Universidade do Minho
Escola de Engenharia

Nuno Constantino Castro

Time Series Motif Discovery

June,2012

Nuno Constantino Castro **Time Series Motif Discovery**
UMinho | 2012

Universidade do Minho
Escola de Engenharia

Nuno Constantino Castro

Time Series Motif Discovery

Doctoral Program in Computer Science (MAP-i)

Supervisor:
Doutor Paulo Jorge Azevedo

June, 2012

To Daniela.

Acknowledgments

It has been an incredible journey. But the path was set for me and I just needed to walk throught it. I wish to thank all the people that helped me walk the path.

To begin with, I would like to thank my supervisor Professor Paulo Azevedo. Your support, guidance and encouragement were determinant for the completion of this thesis. The confidence you showed in my ideas inspired me to always attempt to go one step farther.

I would like to acknowledge the University of Minho and all the teachers that have contributed to my education. Also, the Distributed Systems Laboratory, who welcomed me to their noisy group and has been my workplace during the PhD work.

I acknowledge the Fundação para a Ciência e Tecnologia (FCT) for paying my bills through grant SFRH / BD / 33303 / 2008.

I also wish to thank to all my friends. Your support and motivation are indeed a priviledge.

I would also like to thank my parents. Perhaps you cannot read the English in this thesis, but you're in every single word of it.

Finally, I would like to thank my beloved wife Daniela. My inspiration comes from you in every way.

Abstract

Time series data are daily produced in massive proportions in virtually every field. Most of the data are stored in time series databases. To find patterns in the databases is an important problem. These patterns, also known as motifs, provide useful insight to the domain expert and summarize the database. They have been widely used in areas as diverse as finance and medicine. Despite there are many algorithms for the task, they typically do not scale and need to set several parameters. We propose a novel algorithm that runs in linear time, is also space efficient and only needs to set one parameter. It fully exploits the state of the art time series representation (SAX – Symbolic Aggregate Approximation) technique to extract motifs at several resolutions. This property allows the algorithm to skip expensive distance calculations that are typically employed by other algorithms. We also propose an approach to calculate time series motifs statistical significance. Despite there are many approaches in the literature to find time series motifs efficiently, surprisingly there is no approach that calculates a motifs statistical significance. Our proposal leverages work from the bioinformatics community by using a symbolic definition of time series motifs to derive each motif's p-value. We estimate the expected frequency of a motif by using Markov Chain models. The p-value is then assessed by comparing the actual frequency to the estimated one using statistical hypothesis tests. Our contribution gives means to the application of a powerful technique - statistical tests - to a time series setting. This provides researchers and practitioners with an important tool to evaluate automatically the degree of relevance of each extracted motif. Finally, we propose an approach to automatically derive the Symbolic Aggregate Approximation (iSAX) time series representation's parameters. This technique is widely used in time series data mining. Its popularity arises from the fact that it is symbolic, reduces the dimensionality of the series, allows lower bounding and is space efficient. However, the need to set the symbolic length and alphabet size parameters limits the applicability of the representation since the

best parameter setting is highly application dependent. Typically, these are either set to a fixed value (e.g. 8) or experimentally probed for the best configuration. The technique, referred as AutoiSAX, not only discovers the best parameter setting for each time series in the database but also finds the alphabet size for each iSAX symbol within the same word. It is based on the simple and intuitive ideas of time series complexity and standard deviation. The technique can be smoothly embedded in existing data mining tasks as an efficient sub-routine. We analyse the impact of using AutoiSAX in visualization interpretability, classification accuracy and motif mining results. Our contribution aims to make iSAX a more general approach as it evolves towards a parameter-free method.

Resumo

As séries temporais são produzidas diariamente em quantidades massivas em diferentes áreas de trabalho. Estes dados são guardados em bases de dados de séries temporais. Descobrir padrões desconhecidos e repetidos em bases de dados de séries temporais é um desafio pertinente. Estes padrões, também conhecidos como motivos, dão uma nova perspectiva da base de dados, ajudando a explorá-la e sumarizá-la. São frequentemente utilizados em áreas tão diversas como as finanças ou a medicina. Apesar de existirem diversos algoritmos destinados à execução desta tarefa, geralmente não apresentam uma boa escalabilidade e exigem a configuração de vários parâmetros. Propomos, neste trabalho, a criação de um novo algoritmo que executa em tempo linear e que é igualmente eficiente em termos de memória usada, necessitando apenas de um parâmetro. Este algoritmo usufrui da melhor técnica de representação de séries temporais para extrair motivos em várias resoluções (SAX). Esta propriedade permite evitar o cálculo de distâncias que têm um custo computacional muito elevado, cálculo este geralmente presente noutros algoritmos. Nesta tese também fazemos uma proposta para calcular a significância estatística de motivos em séries temporais. Apesar de existirem muitas propostas para a detecção eficiente de motivos em séries temporais, surpreendentemente não existe nenhuma aproximação para calcular a sua significância estatística. A nossa proposta é enriquecida pelo trabalho da área bioinformática, sendo usada uma definição simbólica de motivo para derivar o seu respectivo p-value. Estimamos a frequência esperada de um motivo usando modelos de cadeias de Markov. O p-value associado a um teste estatístico é calculado comparando a frequência real com a frequência estimada de cada padrão. A nossa contribuição permite a aplicação de uma técnica poderosa, testes estatísticos, para a área das séries temporais. Proporciona assim, aos investigadores e utilizadores, uma ferramenta importante para avaliarem, de forma automática, a relevância de cada motivo extraído dos seus dados. Por fim, propomos uma metodologia para derivar de forma automática os parâmetros da

representação de séries temporais Symbolic Aggregate Approximation (iSAX). Esta técnica é vastamente utilizada na área de Extracção de Conhecimento em séries temporais. A sua popularidade surge associada ao facto de ser simbólica, de reduzir o tamanho das séries, de permitir aproximar a Distância Euclidiana nas séries originais e ser eficiente em termos de espaço. Contudo, a necessidade de definir os parâmetros comprimento da representação e tamanho do alfabeto limita a sua utilização na prática, uma vez que o parâmetro mais adequado está dependente da área em causa. Normalmente, estes são definidos quer para um valor fixo (por exemplo, 8). A técnica, designada por AutoiSAX, não só extrai a melhor configuração do parâmetro para cada série temporal da base de dados como consegue encontrar a dimensão do alfabeto para cada símbolo iSAX dentro da mesma palavra. Baseia-se em ideias simples e intuitivas como a complexidade das séries temporais e no desvio padrão. A técnica pode ser facilmente incorporada como uma sub-rotina eficiente em tarefas existentes de extracção de conhecimento. Analisamos também o impacto da utilização do AutoiSAX na capacidade interpretativa em tarefas de visualização, exactidão da classificação e na qualidade dos motivos extraídos. A nossa proposta pretende que a iSAX se consolide como uma abordagem mais geral à medida que se vai constituindo como uma metodologia livre de parâmetros.

Contents

List of Tables

List of Figures

List of Algorithms

- 1 -

Introduction

1.1 Background

Data Mining or Knowledge Discovery in Databases (KDD) is an important area of computer sciences. The relevance of this area is due to the enormous amount of information daily produced in different sources, e.g. the web, biological processes, finance, the aeronautic industry, retail, and telecommunications. A considerable percentage of this information represents temporal events which are typically stored in the form of time series. For example, it was shown that more than 75% of newspaper images were time series charts, using a random sample of 4,000 graphics from 15 popular newspapers published from 1974 to 1980 (Tufte and Howard, 1983).

Examples of time series are the variation of a stock index, the consumption of a certain good, the daily blood pressure of an individual, the annual rainfall in a city, the number of web page visits per second, the brain electrical activity of a patient measured at 256 Hz in an electroencephalogram (EEG), the motion of a blob in a football player's right foot as he shoots the ball, the daily evolution of the oil price, or the prime minister popularity over time. Other types of data such as DNA or video are not time series in their raw format, but they can be converted to time series. This transformation enables the use of a large number of algorithms specifically tailored to time series in other types of data. Figure 1.1 shows several examples of real world time series, compiled from a variety of Eamonn Keogh's papers (Keogh, 2006). Figure 1.2 shows one technique that can be used to convert a leaf image to a time series.

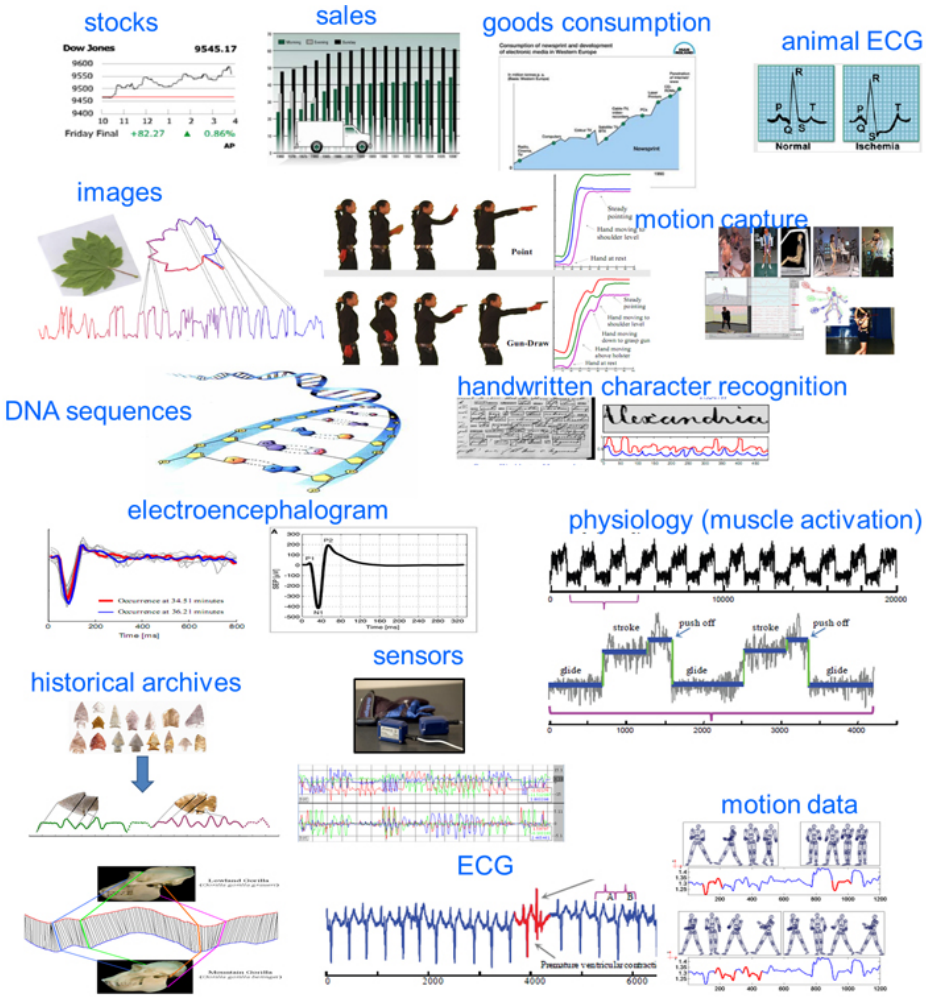


Figure 1.1: Examples of time series (from Keogh (2006))

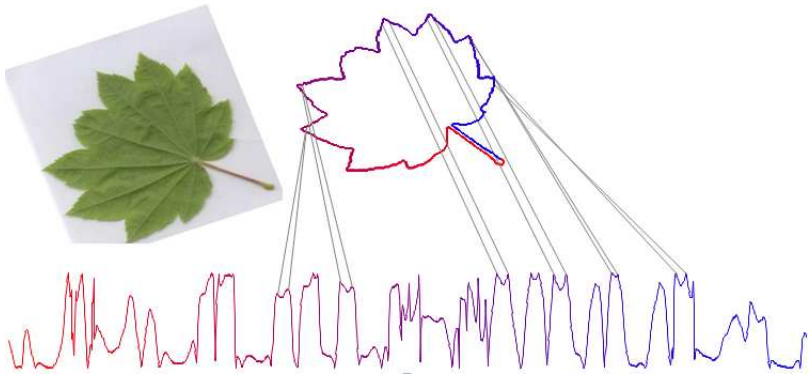


Figure 1.2: Conversion of a leaf image to time series (from Keogh (2006))

1.2 Motivation

Time series data exhibit characteristics that make their analysis particularly difficult. The large amount of data derived for them leads any attempt for manual data analysis to failure. Automatic processing is also hard because we are typically dealing with massive datasets. For example (Keogh, 2006), one hour of an electrocardiogram can produce one gigabyte (GB) of data, a typical weblog 5GB a week, and some existing databases such as the MACHO database (Alcock et al., 1997) have terabytes of information, growing at a gigabyte daily rate. Another characteristic is that we are often dealing with subjectivity, as time series similarity (the core of many data mining tasks) depends on the user, domain, and task (Keogh, 2006). For example, what is an important pattern in EEG can be considered meaningless in motion data. Also, time series usually present different data formats, sampling rates, missing values and noise, which makes the task even more complicated. However, there are several phenomena expected to be identified among databases of this type, namely through pattern discovery, classification, clustering, query by content, abnormality detection, and forecast of property values.

The task of finding previously unknown frequent patterns in time series is of particular importance. Also known as "recurrent patterns", "typical shapes", "similar patterns", or just "motifs" (Lin et al., 2002), these time series subsequences usually describe the time series at hand, providing invaluable insight to the domain expert (Ferreira et al., 2006). For example, in EEG time series a motif may be a pattern that usually precedes a seizure; in DNA it may be a sequence that has been preserved through evolution (Ferreira et al., 2006); in motion capture it may be a particular gesture (e.g. throw ball); in insect images, it may be a peculiar characteristic on the insect's wing that is only common to the animals in the same species; in music, it may be a specific sequence of rhythms; in telecommunications, a burst in traffic which typically happens when major social events are located near an antenna. These tasks can also be encapsulated in other data mining techniques such as finding association rules in time series or forecasting.

Since their introduction in Lin et al. (2002), several motif discovery algorithms have been proposed (chapter 3 contains a survey). However, these algorithms are limited in at least one of the following:

- They present a large computational complexity. As time series datasets are typically massive, quadratic complexity algorithms are not suitable for the task. They can only be effectively applied to small examples;
- They are disk innefficient. Typical algorithms use random disk accesses, while it is known that sequential accesses can be up to 90% faster, considering the access to the same amount of data;
- They are space innefficient. To tackle the disk efficiency issue, the entire dataset is loaded to main memory. This is unfeasible for large datasets;
- They are tailored to univariate time series. Many important domains (e.g. EEG and sensor networks) present more than one time series for the same time interval (multivariate time series);
- They present several parameters. Despite parameters are useful to enable the algorithm to work in many different application, they are typically unintuitive to optimize even by the domain expert. For example, the number of columns of matrix $\hat{S}$, the size of the sliding window, the minimum intra-motif distance threshold (motif range R), alphabet cardinality, minimum support, etc. These kinds of parameters are highly dependent on the dataset at hand, and must be tuned by testing several parameter configurations and experimentally selecting the best one. This is untenable even for datasets of moderate size, as the algorithm needs to be run multiple times to search for the best parameter setting;
- They are not suitable for streaming data, which are continuously derived and must be processed in a real-time fashion. Streaming time series are particularly important, as many application domains continuously derive data which need to be mined instantaneously. The algorithms required to analyse these types

of data have specific characteristics: they can only see each data point once, cannot store the full dataset, and need to continuously adapt the learned model to the arriving data stream;

- They lack objective evaluation measures for the discovered motifs. Motif discovery is an unsupervised task. Motifs are not previously labeled (as in classification). To evaluate the discovered motifs in terms of meaning or correctness is not trivial, as the evaluation is subjective. The best way to evaluate motifs according to its meaning or correctness is to have a domain expert to perform such task. However, this is untenable in practice, as we want the motif discovery process to be autonomous. Also, the number of motifs that are returned by any motif discovery algorithm can be prohibitively large, due to the algorithm, dataset or user parameters (Ferreira and Azevedo, 2007). In other areas such as bioinformatics, statistical or information measures are used. To the best of our knowledge, an evaluation measure to rank motifs in terms of significant does not exist in the literature.

1.3 Goals

In this thesis, we investigate the time series motif discovery problem by tackling the above described limitations of existing algorithms. Particularly, we aim to defend the following statement:

We believe that the state of the art in motif discovery can be improved, by developing scalable algorithms with fewer parameters. To further increase the understanding in time series mining, we aim to develop a methodology to automatically evaluate the discovered motifs. We hope that this approach has significant impact in the time series data mining community.

1.4 Contributions

The contributions of this thesis can be summarized as follows:

- We present a thorough survey of the state of the art in time series motif discovery. To the best of our knowledge, a comprehensive survey of this area is not yet available in the literature;

- We provide a scalable algorithm to retrieve approximate motifs from time series databases. We achieve time efficiency by using a single sequential disk scan to read the time series database, a clever time series representation and constant complexity motif counting technique. The single sequential disk pass makes the method a strong candidate for data streaming applications. Memory efficiency is achieved by combining our method with the space-saving algorithm (Metwally et al., 2005), now applied to time series data mining. Our approach is based on the state of art time series representation – iSAX (Shieh and Keogh, 2008), exploiting its multiresolution property to derive motifs at different resolutions. We leverage this property to avoid expensive distance calculations in the raw data. It also enables the development of powerful visualization applications, allowing "drill-down" and "roll-up" like navigation in the Top-K motifs hierarchy structure. We provide the iMotifs Visualization Tool as an example of such hierarchical explorer. To be able to visually mine the motifs yields better understanding and intuition about the database at hand to the user;

- We propose an approach to calculate time series motifs statistical significance. Our proposal leverages work from the bioinformatics community by using a symbolic definition of time series motifs to derive each motif's p-value. We estimate the expected frequency of a motif by using Markov Chain models. The p-value is then assessed by comparing the actual frequency to the estimated one using statistical hypothesis tests. Our contribution gives means to the application of a powerful technique – statistical tests – to a time series

setting. This provides researchers and practitioners with an important tool to evaluate automatically the degree of relevance of each extracted motif;

- We propose an approach to automatically derive the iSAX time series representation's parameters. The technique, referred as AutoiSAX, not only discovers the best parameter setting for each time series in the database but also finds the alphabet size for each iSAX symbol within the same word. It is based on the simple and intuitive ideas of time series complexity and standard deviation. The technique can be smoothly embedded in existing data mining tasks as an efficient sub-routine. We analyse the impact of using AutoiSAX in visualization interpretability, classification accuracy and motif mining results. Our contribution aims to make iSAX a more general approach as it evolves towards a parameter-free method.

This thesis generalizes our work from the following publications: Castro and Azevedo (2010, 2011, 2012a,b).

1.5 Document Structure

This thesis is organized as follows. In chapter 2, we present an overview to time series data mining. We survey the related work in time series motif discovery in chapter 3. In chapter 4, we propose an efficient motif discovery algorithm. We present our approach to assess the motif statistical significance in 5. Chapter 6 presents an iSAX parameter estimation approach. Finally, we briefly summarize our thesis in chapter 7.

- 2 -

Time Series Data Mining

In this chapter we perform an overview of the overall state of the art in time series data mining. This chapter is of utmost importance to contextualize this thesis and make it self-contained. It follows Keogh (2006) in structure of exposition and provides essential background for the understanding of this thesis. First, we introduce data mining and its most common tasks in section 2.1. Then, in section 2.2, we introduce basic time series concepts. We review the most common time series data mining tasks in section 2.3. Time series similarity is dealt in section 2.4, pre-processing is covered in 2.5, and time series approximate representations in 2.6. Finally, in 2.8 we briefly discuss algorithm's parameters.

2.1 Data Mining

Data Mining is a relatively contemporary area of computer science. It can defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" (Piatetsky-Shapiro and Frawley, 1991). Data Mining is also known by other terms having the same or a marginally different meaning, such as "data dredging" (Han and Kamber, 2000), "knowledge extraction", or "knowledge discovery in databases" (KDD) (Piatetsky-Shapiro, 2006). This area combines knowledge from several other computer science fields, as Artificial Intelligence, Machine Learning, Statistics, Databases, Information Theory, Information Retrieval, Visualization, among others. Data Mining tackles problems using both theoretical-based approaches (e.g. Statistics) and heuristic ones (Machine Learning).

Due to the ever increase amount of raw data available in industrial and scientific databases, Data Mining appeared in the early 80's, with the goal of "making sense" of the large amount of data available (Piatetsky-Shapiro, 2006). Although existing systems could analyse the stored data, they could not rapidly retrieve aggregated data, e.g. Online Transaction Processing (OLTP). Others, such as Online Analytical Processing (OLAP) provided more complex operations such as aggregations, "drill-down" or "roll-up", and "slicing and dicing". Despite more powerful, these were constrained by user-driven actions (querying). A powerful framework to find implicit information not previously known by human operators was necessary. As it is often said in conferences of this area, "the goal of data mining is to provide answers to questions we do not know yet".

The huge volume of data available brings some other problems to the table. Apart from the fact that the large dimension of data is a problem on its own, the data is also often noisy, incomplete and imprecise. To cope with the problems existing in "real world" data, it is usually necessary to have a pre-processing phase, before the actual knowledge extraction phase. Albeit people often refer to "Data Mining" as the whole knowledge extraction process, it is only an intermediary step. The entire routine is as follows:

1. Pre-processing:

   - Data Cleansing - Removal of noise, inconsistent data and missing;
   - Data Integration - Combination of multiple data sources;
   - Data Transformation - Set of operations effectuated for easing the extraction process, like data transformation and aggregation.

     It is noteworthy that in *Data Warehousing* systems, these steps are implicitly handled. In fact, these steps resemble the *Extraction, Transformation and Loading* (ETL) stage characteristic of those systems.

2. Data Mining - Fundamental stage where a variety of methods, tools and techniques are applied to extract structure and meaning from the data;

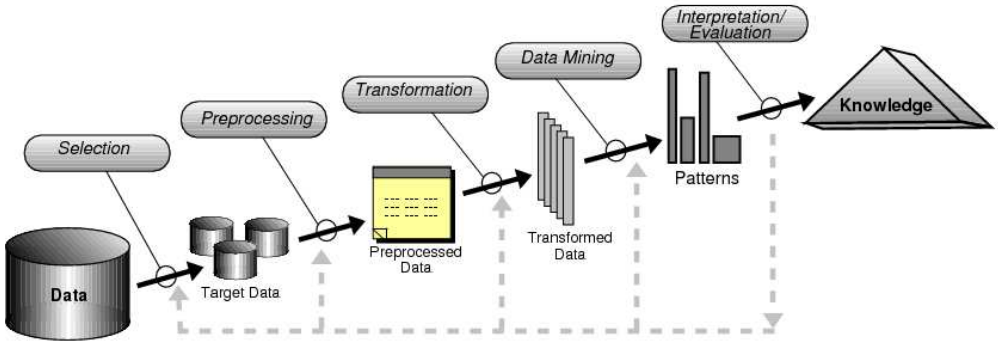


Figure 2.1: Representation of the Knowledge Extraction Process (from Fayyad et al. (1996))

3. Post-Processing:

   - Knowledge Verification - Discovered information is checked for interestingness by experts or using an automatic method;
   - Knowledge Visualization - The final results of the complete knowledge extraction process is presented, often graphically, to the end user;
   - Knowledge Deployment - Discovered information is put into action.

The steps described are not mandatory in the knowledge extraction process. Some may be merged or removed without affecting the quality of the process, depending on the specific characteristics of the application at hand. Figure 2.1 represents the knowledge extraction process.

2.1.1 Data Mining Techniques

The actual Data Mining process is comprised by several techniques (or tasks). These include classification, clustering, associations discovery, anomaly detection, forecasting, among others. Classification is the task of learning a method to classify new instances, from a pre-labeled set of instances. An instance is constituted by several attributes. Instance features can be of several types (numerical, categorical). A classifier algorithm can be based on a set of rules, decision trees, neural networks, support vector machines, statistic methods, distance measures, etc. Clustering

methods find the "natural" groups of instances given unlabeled data. There are various approaches and algorithms to find the different clusters. Each one is based on a similarity measure which defines the proximity of the instances based on a specific common characteristic (e.g. Euclidean Distance). Association Rules Discovery is the task of finding existing relations or patterns in the data. Association normally takes into account the frequency a set of items appear in the data. For example: items $A\&B\&C$ appear frequently together. Generally these relations are presented in the form of patterns or association rules. Anomaly detection is the process of identifying states of a certain system that considerably differ from its normal behavior (Penedones, 2005). Forecasting intends to predict the future state of a variable, given its past states.

2.1.2 Data Mining application areas

Data Mining techniques can be applied to a broad range of areas that range from science to the industry. In the Web it can be applied to the huge amount of search engine stored data, it can help the government and other institutions to detect fraudulent activities and loan-risk profiles in banks. It has vast relevance in telecommunications, banking, finance, and retail. One of the most well known data mining stories refers to a famous american chain of supermarkets. Through intensive analysis of the transactions of goods bought over a period of time, analysts found that beer and diapers were often bought together. On Friday afternoons, young men who bought diapers would also likely buy beer. This result could hardly be predicted, as it was not likely that someone queried the system with such question. This is an good example of the difference between data mining and querying. The reason for that was extrapoled to be that young parents went shopping for diapers and also bought beer. The retail experts placed the highest profit beer with the highest profit diapers and the sales increased. Despite this story has been recently claimed to be an urban legend (Whitehorn, 2010), it highlights the industrial edge that can be achieved by using data mining techniques.

2.1.3 Temporal Data Mining

Temporal data mining is the mining process that is related with a temporal aspect (Mörchen and Ultsch, 2007). A large number of areas and methods are covered by this type of data mining. This is not limited to treating the temporal aspect as another static variable and using traditional techniques (Mörchen and Ultsch, 2007). A large number of algorithms and techniques were developed specifically for these types of data. Temporal data can be defined as an ordered sequence of observations - time points - that represent the variation of a property over time. If the nature of the variable is categorical, nominal, or discrete, then they are called temporal sequences or symbolic time series. If each time point is numeric then the time series is called a numeric time series. We focus only on numeric time series. For the remainder of this thesis, we will refer to numeric time series just by *time series*. A great variety of phenomena can be described using a time series: temperature, stocks, energy – every variable that is a part of a measurable process that occurs along time. Most Data Mining tasks can be applied to time series: classification, clustering, associations, forecasting, abnormality detection, and frequent pattern discovery. This is the type of Data Mining techniques that we propose to study and develop in our work.

2.2 Time Series Introduction

A time series can be formally defined as follows:

Definition 2.1. (Time series) A time series T of length n is an ordered succession of observations of a variable $(t_1, \ldots, t_n)$ over time, with $t_i \in \mathbb{R}$.

Figure 2.2 shows an example of a time series from an Electrocardiogram (ECG).

Data mining applications typically use subsections of the time series, rather than the whole time series. For example, to find the most frequent subsequence of length 100 in the time series database.

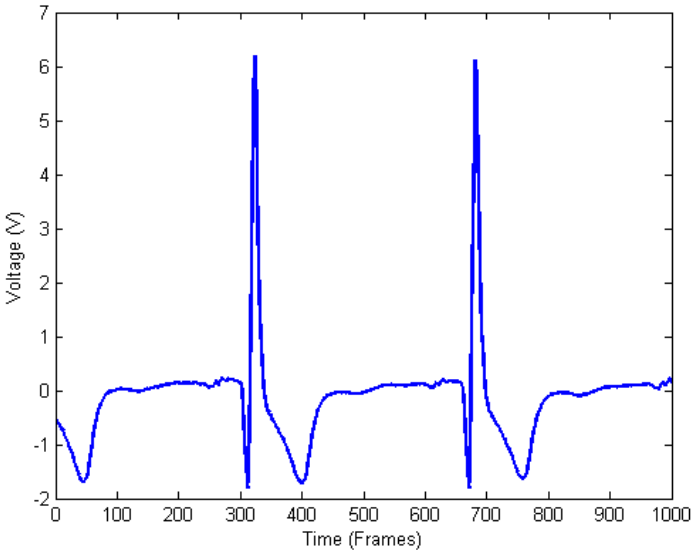


Figure 2.2: Example of a ECG time series

Definition 2.2. (Time series subsequence) Given a time series T of length n, a subsequence $S = (s_i, ..., s_{i+m-1})$ is a sampling of $m \leq n$ contiguous positions of T, such that $1 \leq i \leq n - m + 1$ (Lin et al., 2002).

In figure 2.3 we represent an example of a time series subsequence of length 80 (red) of the previously displayed time series.

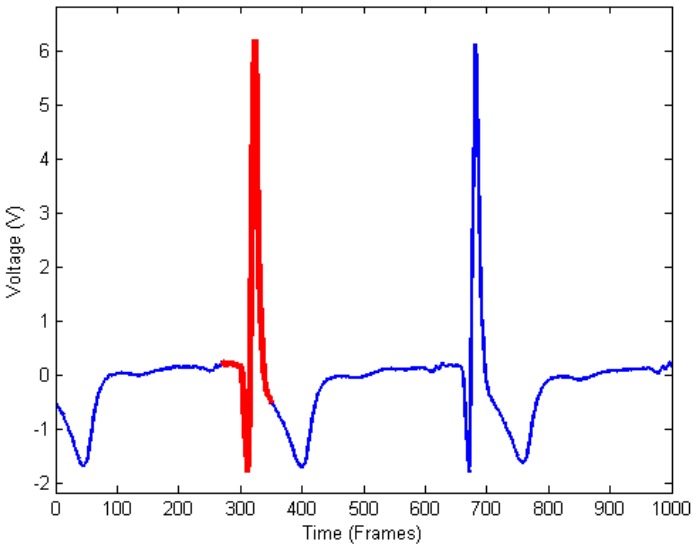


Figure 2.3: Example of a time series subsequence

Time series can contain more than one single variable changing through time. For example, an EEG recording with several electrodes placed in the scalp over the period of one minute.

Definition 2.3. (Multivariate time series) A Multivariate time series is a set of several time series (variables) in the same time range.

A time series can be generated through a device or process that is continuously deriving data. In this specific case it is called streaming time series, or a data stream. We now formalize this concept.

Definition 2.4. (Streaming time series) A streaming time series X is a time series with $n = \infty$, whose data points arrive continuously at an arbitrary rate.

This type of time series is present in many application domains e.g. data captured by sensor networks. In this case we may be in the presence of multivariate streaming time series. Typically, researchers transform streaming time series into static time series by defining an end point to the time series (length n to a specific value). Then, a traditional offline data mining algorithm i.e. an algorithm where the time series does not change during its execution, is applied to the truncated time series. Online (or real-time) algorithms go beyond this approach and can be applied to streaming data. As soon as a new data point is available the internal state of the algorithm is updated according to this new point.

It is important to distinguish one notion - that of anytime algorithm - which is independent from the notions of online or offline algorithm. An anytime algorithm is an important type of algorithm in data mining, which trades performance in execution time for performance in terms of results (Grass and Zilberstein, 1996; Ye et al., 2008). The algorithm can be interrupted at anytime, and it always presents the best-so-far answer available (Ye et al., 2008), whose quality improves monotonically over time. Let us consider the following intentionally exaggerated example. We have a set of 1000000 different univariate time series, and want to find the two nearest neighbors, i.e. the two time series that are most similar to each other in

terms of Euclidean Distance (see section 2.4). We can have a batch (non-interactive) algorithm that executes during one hour and then returns the optimal result. Using an anytime algorithm, we can at any time (thus the name) interrupt the algorithm to consult the best-so-far result and then resume the execution. It is expected that the best so far is a good approximation of the correct result.

2.3 Data Mining Tasks in Time Series

Time series are commonly analysed using classical or traditional data mining techniques. Although we do not follow these approaches, we briefly introduce them for completeness.

Classical approaches are techniques that model the time series data using statistics. These techniques typically make certain assumptions. For example, they assume the time series can be linearly modeled and is stationary. That is, they assume that the behavior of the time series does not change through time. These approaches need to transform the time series by removing some of its characteristics such as trend, variance, or seasonality. Examples of these methods include moving averages (MA), exponential moving averages, auto regressive models (AR), and a combination of moving averages with auto regressive models (ARMA). More sophisticated models such as the ARIMA and Holt-Winters handle seasonality, but they can not handle more complex series. Despite the strong assumptions of these approaches, they present relatively good results with time series that can be linearly modeled.

Traditional data mining techniques are the set of algorithms and techniques that are applied in machine learning to general data. That is, they are not tailored to specific types of data. Artificial Neural networks (ANN) (Mitchell, 1997) and Support Vector Machines (SVM) (Cortes and Vapnik, 1995) are amongst the most used in the literature to accomplish these tasks.

These techniques can be applied to time series data by using, for example, time delay embedding. A window of fixed size is slid over the time series, and each window is

treated as an input vector to train the system. The idea is to view the time series as a dynamic system. For example, in a classification scenario, one can train an ANN with windows (subsequences) of the different classes.

We follow the approach in Keogh (2006), where time series data mining is considered a separate area due to the peculiarities of these types of data. In this section we briefly describe the data mining tasks that are commonly applied to time series data. These include classification, clustering, motif discovery, association rule discovery, query by content, visualization, and anomaly detection.

Classification

There are two types of classification in time series mining (Mörchen and Ultsch, 2007): classifying whole time series, and classifying each individual data point. In the former, given a set of labeled (pre-classified) time series, the goal is to learn a model to classify new (unlabeled) time series. For example, given a set of electrocardiogram (ECG) recordings, each one comprising an individual time series, to classify new ECG as normal or abnormal. At another example application, given a set of motion capture time series of different persons pre-labeled as young, adult or old, to classify new instances of motion capture time series in one of the three classes. In the latter, there are labels for each time point. The classifier learns from the time points of the training set, and given a new time series, the task is to classify each point in one of the classes. For example, a time series containing the number of visits per hour to a web site, and a class for each point: normal use, heavy use, attack. The classifier is trained over the labeled training set, and then its goal is to classify new test points in one of the three labels. If the system is configured to work online (see section 2.2), the test points arrive to the system in a streaming fashion. The task of time point classification task is strongly connected to prediction and abnormality detection (Mörchen and Ultsch, 2007).

The state of the art in classification algorithms for time series is dominated by approaches that use time series representations suitable to traditional data mining

classifiers. Few methods exist that are designed explicitly for time series (Mörchen and Ultsch, 2007). However, it has been found that using a simple 1-NN approach with EDis surprinsingly effective (Keogh et al., 2009). Time series individual point classification is not a common explicit research topic. This type of classification is usually tackled implicitly in prediction or novelty detection tasks.

Clustering

Clustering time series is the process of grouping similar time series according to a similarity measure (e.g. Euclidean Distance). It can be viewed as the unsupervised version of classification, because instances are not previously labeled with a class. The actual goal of clustering is to find the natural classes or groups in the data, and then place each instance (time series) in the groups. Time series clustering results are often illustrated by dendrograms. These tree diagrams graphically represent the result of a hierarchical clustering algorithm. Figure 2.4 shows the results of hierarchical clustering over nine time series in a dendrogram.

These illustrations are frequently used as subjective validators of novel similarity measures. If we apply clustering and it does not provide an intuitive dendrogram, then the similarity measure can be discarded (Keogh, 2006). Clustering results depend heavily on the choice of a proper similarity measure (Mörchen and Ultsch, 2007).

Subsequence clustering is the technique of clustering the subsequences extracted from a single long time series using a sliding window. This technique has empirically shown to produce meaningless results (Keogh and Lin, 2005), because overlapping subsequences are typically used.

Motif discovery

Finding repeated subsequences, frequent patterns, or motifs in time series is the task of searching for previously unknown recurrent patterns in time series. Despite there

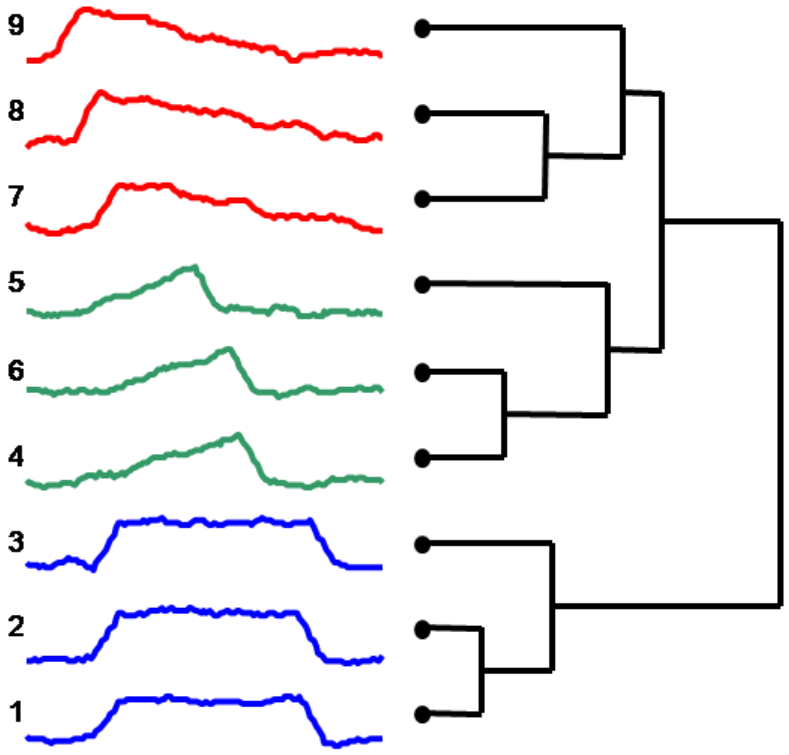


Figure 2.4: Example of a time series dendrogram (from Keogh (2006))

is a large volume of research in searching for known patterns in time series (query by content), to search for unknown patterns is a relatively recent research problem. It was proposed in Lin et al. (2002).

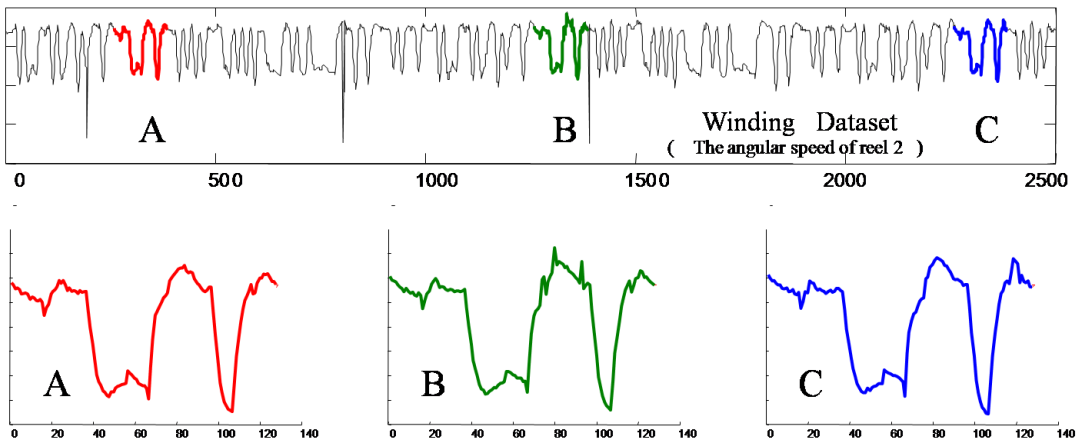


Figure 2.5: The task of motif discovery (from Keogh (2006)). Notice that the discovered subsequences are not identical.

There are two types of motif discovery: univariate and multivariate. Univariate

motif discovery aims to find repeated subsequences in one single, typically longer, time series; or one database of time series. The goal of multivariate motif discovery is to find motifs that span across different time series for the same time range. It is believed that the most frequent motifs of a time series can help represent/summarize the time series, or give valuable insight to the domain expert about the problem under study (Ferreira et al., 2006). However, for the representation to be valid we need to guarantee that the patterns are meaningful, which remains unclear. Motif discovery is our main research topic and will be covered in depth in the remaining chapters.

Association Rules

In the global data mining context, association rules discovery is the task of finding existing relations or patterns in the data. In time series data mining this task is similar: to find relations or patterns between different attributes of the time series. To properly define these attributes is a crucial step in association rule mining. For example, if the attributes are time series segments, we can reach the following association rule: two peaks following plateau following another peak leads to an increasing trend with probability 0.9. Association rules are interpretable by humans because they are close to natural language. For that reason, experts can gain valuable knowledge about the application domain.

Visualization

The task of visualization in data mining is related to the fact that the human eye is the best known pattern finder. These techniques aim to explore this fact and visually present the data using graphical objects that can easily be interpreted by humans. One important aspect is to present a large amount of data in a condensed space (the size of the computer screen). That is, to present intuitively truly massive datasets. It is surprising that visualization is typically limited to the common two-

axis approach, instead of more powerful visualization techniques, which have not yet been properly addressed (Mörchen and Ultsch, 2007).

Query by Content

Query by content is the task of retrieving the most similar time series in the database (best match) to the query time series. One can also search for subsequences instead. This retrieval is often performed with indexing techniques in place to speed up the search. It was one of the first research topics in data mining and is present in many application domains: to find the most similar image to a given image from a database (recall that an image can be converted to a time series); find the best match of a given DNA sequence; find at which day the stocks behaved most closely related to today's behavior; find the recording that is most similar to a seizure (given as query) in the EEG recordings database. This task is related to subsequence matching (see section 2.4).

Anomaly detection

Anomaly detection, abnormality detection, or novelty detection is the task of detecting abnormal behavior of a time series, following a certain "normal" criterion. This task contrasts with the task of motif discovery, as we are trying to find the least frequent pattern in the series. The criteria are usually related to how unexpected, surprising or interesting a pattern is, which is heavily dependent on the application domain at hand. Examples of applications are intrusion detection, detecting abnormal segments of an ECG, outlier detection (as pre-processing), detecting anomalies in a space shuttle lunch, fraud in credit card, telecom or bank loans, among many others.

2.4 Similarity

At the core of all the time series data mining tasks described in the previous subsection is similarity. In classification, we often search for the time series example that is most similar to the instance we want to classify. In clustering, groups of time series are formed according to some similarity that the elements of each single group of time series share. That is, examples in a group or cluster are similar according to a certain similarity criterion. In motif discovery, we need to have a notion of similarity to consider that one subsequence repeats over time. The concept of similarity is inherent to the notion of repetition: we need to count similar (not identical) subsequences. In query by content, we search for the best match of a given query, where the best match is the most similar time series subsequence. In anomaly detection, we find the least similar subsequences, and so forth. Similarity is, according to the dictionary, "the state of being similar; likeness; resemblance". This concept is hard to quantify even for humans, as there is a considerable amount of subjectivity involved. We take a more pragmatic approach in order to be able to quantify similarity between two time series, or similarity matching. Hence, we define similarity matching as:

Definition 2.5. (Similarity matching) Given time series query Q, a time series database C and a similarity measure D, find the $C_i \in C$ that best matches Q according to D (Keogh, 2006).

In data mining we are also interested in finding the most similar subsequence to a given subsequence in a time series. This notion is expressed in definition (2.6).

Definition 2.6. (Subsequence matching) Given time series subsequence query Q, a database C and a distance measure D, find the location i in C that best matches Q according to D (Keogh, 2006).

The database is usually obtained by retrieving all subsequences of a time series, by sliding a window of size $|Q|$ incrementally across the time series. Figure 2.6 visualizes this process.

Figure 2.6: Retrieval of all subsequences of a time series using a sliding window (from Rebbapragada (2007))

When we refer to similarity, we are often measuring its opposite: dissimilarity. This notion is typically expressed using a distance measure.

Definition 2.7. (Distance measure) Given two objects O_1 and O_2 the distance between them is the dissimilarity between the objects and is denoted by $D(O_1, O_2)$ (Keogh, 2006).

Distance measures should satisfy several properties (Keogh, 2006) to be a metric:

- Non-negativity: $D(X, Y) \geq 0$;
- Symmetry: $D(X, Y) = D(Y, X)$;
- Constancy: $D(X, X) = 0$;
- Triangular Inequality: $D(X, Z) \leq D(X, Y) + D(Y, Z)$.

The intuition behind the triangular inequality is: if one time series X is similar to another one Y, and Y is similar to Z, we have the guarantee that X is similar to Z. This is an important property of distance measures, because it can save a large number of calculations in the similarity search process. For example, we are searching for the closest time series to Q in C and we are using the d_w distance measure holding the triangular inequality: if we know $d_w(Q, c_a)$ and $d_w(c_a, c_b)$, we may skip the calculation of $d_w(Q, c_b)$ because with the triangular inequality property we can upper bound it to $d_w(Q, c_a) - d_w(c_a, c_b)$. If this upper bound is greater than the best-so-far time series, we can skip the expensive $d_w(Q, c_b)$ calculation (Keogh, 2006). The symmetry property can also be used for preventing us to do about half the distance calculations. We can store the value of D(X,Y) and then re-use it when

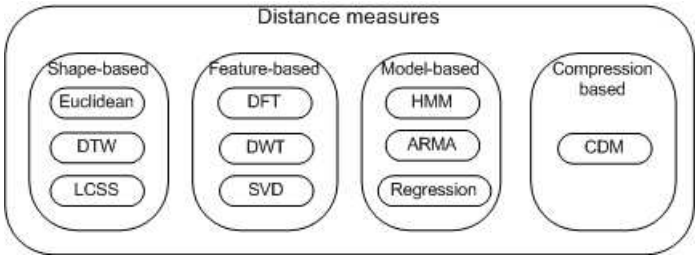


Figure 2.7: Categorization of time series distance measures (based on Mörchen and Ultsch (2007))

we need to know D(Y,X) (Lin et al., 2002). However this requires sufficient storage space, which can be problematic even for moderate size datasets.

There are four categories for similarity measures in time series (Mörchen and Ultsch, 2007): shape-based, feature-based, model-based, and compression-based. Shape-based measures compare time series similarity in terms of shape, or appearance. Feature-based measures extract features that are time-independent and compare these using traditional techniques. Model-based measures fit models to the time series and then compare the models. Compression-based measures compare how similar the compressed version of the time series are. In Fig. 2.7 we depict a categorization of several distance measures.

Shape-based distance measures seem to capture the intuition humans have when comparing time series: they look at the time series and subjectively judge their similarity according to their overall shape proximity. The Euclidean Distance (ED) is by far the most used distance measure to compare numeric time series (Lin et al., 2003). The ED of two time series is simply the square root of the squared sum of the pairwise time series differences (see expression 2.1 with $p = 2$). It is fast to calculate (linear complexity) and has been shown to be trivially indexed in Keogh and Kasetty (2003). Expression 2.1 is the general expression for L_p norms, a generalization of ED for arbitrary p.

$$D(Q, C) = \sqrt[p]{\sum_{i=1}^{n} (q_i - c_i)^p} \tag{2.1}$$

The value of p is the exponent of the differences (in the ED $p = 2$, therefore the differences are squared), which puts different emphasis on large deviations (Mörchen and Ultsch, 2007). L_p norms in general and the ED in particular are very sensitive to:

- Small distortions in the time axis (Keogh and Kasetty, 2003);
- Scaling and offset translation of the amplitudes;
- Noise and outliers Lin et al. (2002);
- Trends.

and they require the two time series to be of the same size in order to perform a one-to-one comparison.

Scaling and translation can produce large distances even if the time series has the same shape (Mörchen and Ultsch, 2007). This problem can be handled by pre-processing the time series. Namely, by applying a z-normalization – removing amplitude scale and offset translation. This issue is covered in detail in section 2.5.

Distortions in the time axis are typically handled by the Dynamic Time Warping (DTW) distance (Berndt and Clifford, 1996). This distance measure takes into account the warping of the time axis of both series in order to better align the two time series.

This measure can be calculated using a dynamic programming approach. It has been empirically demonstrated that DTW outperforms the Euclidean Distance, but is more than two orders of magnitude slower (Keogh and Kasetty, 2003). Nevertheless, it has been recently shown that by using a clever lower-bounding technique, DTW is essentially linear (Ratanamahatana and Keogh, 2005). Noise and outliers can be compensated by: smoothing the time series if the noise is present in the whole time series and not limited to regions; or allowing gap regions that are excluded from the distance calculation. The Longest Common Subsequences (LCSS) (Das et al., 1997)

is an example of such type of method. Trends can also be removed by fitting a line to the time series and then subtracting it to the original time series. Time series that are of different size can be transformed to have the same length by resampling. After that, the distance function can be applied. This technique is also known as Uniform Time Warping (Keogh et al., 2004; Mörchen and Ultsch, 2007). Typically the longer series is down-sampled to the size of the smaller time series. Down-sampling has also been shown to be effective against noise (Argyros and Ermopoulos, 2003; Mörchen and Ultsch, 2007). Despite these limitations, it has been empirically demonstrated in Keogh and Kasetty (2003) using 11 recently published distance measures, that the ED outperforms most of them by one order of magnitude, in terms of classification error rate. Most of the time it is only outperformed by DTW.

Feature-based methods should be used for long time series where shape-based methods do not provide intuitive results (Mörchen and Ultsch, 2007). It is noteworthy that the features to be extracted are domain dependent and are not guaranteed to provide good results. Widely used methods such as Discrete Fourier Transform (DFT) and Discrete Wavelet Transform (DWT) typically transform the original time series into a set of coefficients. Then they select the most important coefficients: those that describe the data according to a certain criterion. For example, DFT coefficients are sorted according to ascending frequency order. Lower frequencies mean slower transitions, whereas higher frequencies mean higher transitions – more variations per period of time. For example, we can smooth a time series by selecting

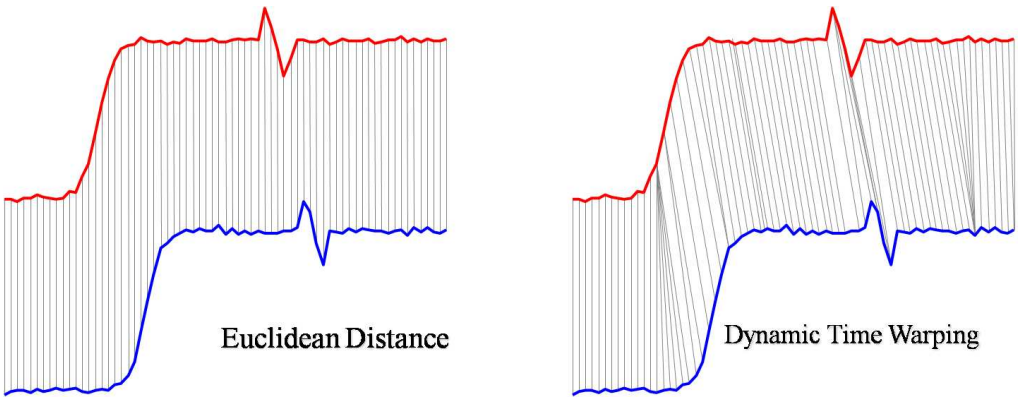


Figure 2.8: Comparison of ED and DTW (from Keogh (2006))

only the first ten DFT coefficients. Obviously, the frequencies or coefficients that are interesting depend completely on the application domain and task at hand. Sudden changes may be interesting in a EEG scan, for example, but meaningless in motion capture where they are most likely to be caused by noise. The coefficients of both series are then trivially compared using the ED or more advanced techniques. Features that are typically used are statistical ones – mean, variance, skewness, ARMA coefficients, parameters of Markov Model, and parameters of Box Jenkins. For further details see Keogh (2006) and the references therein.

Model-based methods should also be used with longer time series. They have the advantage that they incorporate prior knowledge about the process that generates the data (Mörchen and Ultsch, 2007), as they try to model it. The similarity is typically measured by fitting a model to each time series, and then finding the likelihood that one time series was generated by the model of the other time series (Mörchen and Ultsch, 2007). Hidden Markov Models (HMM) are one example of models used in this type of similarity measures.

Compression-based is a rather recent approach (Keogh et al., 2007). The method is the compression based dissimilarity measure (CDM), which uses the Kolmogorov Complexity – the length of the shortest program that is able to generate the given data. Both time series are concatenated and then compressed using off-the-shelf compression algorithms. The compression rate – the size of the compressed concatenation divided by the sum of the individual compressed time series – is then used as a dissimilarity measure. The intuition is that with effective compression algorithms, the simple concatenation of similar time series should present a higher compression rate, whereas very different time series should present lower or no compression (CDM closer to one).

The problem of similarity search is that it is prohibitively expensive. For example, given time series query Q and the ED measure, finding the best match in reference database C using brute-force is found by (Rebbapragada, 2007):

$$min_{1 \leq i \leq |C|} \sum_{t=1}^{|Q|} (C_i(t) - Q(t))^2 \tag{2.2}$$

This can become a problem not only because we need to perform a large number of similarity calculations in our data mining tasks, but most of the time, the results on distance calculations must be stored in main memory. This problem is aggravated by the fact that in most data mining applications we are interested in handling gigabyte or even terabyte sized datasets (as in Yankov et al. (2007b)). It is essential that these calculations are fast. We have several options to make time series similarity search computationally more efficient:

- To use alternate time series representations that either reduce their dimensionality or that have a wealth of existing efficient algorithms available (see section 2.6);
- To search databases faster by: using indexes; or using sequential, rather than random disk access scan (Yankov et al., 2007b);
- To use clever heuristics or explore some properties of metrics such as the triangular inequality to prune the search space;
- To use methods that provide approximate solutions, either probabilistic or anytime algorithms (see 2.2);
- To use new and more efficient similarity measures.

2.5 Pre-processing

To pre-process the time series before the actual data mining process is applied is of utmost importance for the quality of the results. The reason is that raw time series typically present problems:

- Noise and outliers;

- Different file format, sampling rate and size;
- Missing values;
- Distortions;
- Large size.

These factors prevent the most commonly used similarity measures to obtain good results (e.g. Euclidean Distance). The distortions include offset translation, amplitude scaling and linear trends. In this context, "distortions" do not mean that the data is of poor quality, but that the most commonly used distance measures can be mislead by them (Keogh and Pazzani, 1999). Also, time series are usually high-dimensional, i.e. have a large number of data points. A pre-processing step for dimensionality reduction can yield good results. This can be done by choosing clever representations such as DFT, SVD, Wavelets, or SAX (see section 2.6 on representations of time series).

The first transformation to apply is typically the "z-normalization". This transformation removes the offset and scaling effects of the times series by transforming the original time series to have zero mean and standard deviation of one (Keogh and Pazzani, 1999). Offset translation is removed by subtracting to the time series its mean value. Amplitude scaling is removed by dividing the time series by its standard deviation. It has been shown that comparing time series that are not normalized is meaningless (Keogh and Kasetty, 2003). For example, imagine we are analysing three time series representing the distance from the centroid to the contour of three balls: a tennis ball, a football, and a rugby ball. If we apply the ED measure without normalizing first, the football and the rugby ball will have smaller ED, because they are about the same size. The tennis ball will present a higher distance to each of the remaining balls. However, the tennis ball is much more similar to the football, because both are spherical. The normalization of the time series removes the effect of scale and offset, and these two balls present almost zero ED under this setting. In Keogh and Kasetty (2003), the authors criticize the work Kim et al. (2001), that

presents a time series classification method achieving high accuracy. According to them, the authors present biased results caused by overfitting due to the absence of normalization. If we are trying to classify a gun point movement (in motion capture) as a man or woman's, without normalizing the time series, the classifier would easily distinguish between both because of the scaling caused by the difference in height of both genders (Keogh and Kasetty, 2003). However, this result would obviously not generalize. It is noteworthy that whether or not to normalize the time series is a highly domain dependent option. In some application areas, normalization can remove the most interesting parts of the time series. Furthermore, it can remove its discriminative power. Other domains require the focus to be on amplitude order rather than on the absolute value of the series. Amplitude order always calls for normalization. The authors in Keogh and Kasetty (2003) recommend to always normalize, except if there is a strong reason not to do so.

The second transformation to apply is trend removal. This is an important transformation because the effect of linear or higher order trends may greatly affect similarity results. For example, imagine we are comparing the sales of ice-creams in two cities with similar population and climate during one year (Keogh and Pazzani, 1999). If one city's population increases and the other remains steady in that year's time, the ice scream consumption in the latter will present a linear increasing trend, greatly affecting the similarity results (Keogh and Pazzani, 1999). By removing linear trend, we can concentrate our similarity search in the variable we want to analyse. This transformation is performed by linear fitting the time series, and then removing the fitted line from the original time series. Trends can also be removed by finding the first derivative of the time series, i.e. the time series of the differences. However this method amplifies noise (Mörchen and Ultsch, 2007) and should generally not be used. Care should be taken when removing the linear trend, as there are domains where trends are the actual features that contain the useful information.

The third transformation is applied to time series containing noise, where the important feature is the overall shape of the series and not the instant variations it presents. Noise can be removed by smoothing the time series, which can be

accomplished by filtering. Typically, a moving average using a sliding window with a fixed number of data points is used. Noise can also be removed by down-sampling the signal to contain fewer data points. For example, if we have a 10000 points sine wave amplified with random noise, by re-sampling it to 1000 points, the amount of noise in the series would be greatly reduced. Yet another possibility is to apply DFT to the time series and keep only the lower coefficients.

Other transformations include outlier detection, re-sampling, and dimensionality reduction. Outlier detection techniques can also be used in the pre-processing phase to further clean the time series for the tasks at hand. As it was previously explained in section 2.4, time series of different sizes can be compared by re-sampling the longer time series to the size of the shorter, and then by trivially applying a distance measure. Many similarity measures have been proposed in the literature to specifically compare different-sized time series. However, it was empirically demonstrated with a sufficiently large set of experiments that these approaches provide little contribution, as the described problem does not exist (Ratanamahatana and Keogh, 2005). Time series dimensionality can be strongly reduced by using a representation of the time series instead of the original. The relevance of this subject demands deeper investigation. In the context of time series data mining, some algorithms are designed to work with representations instead of the raw data. We proceed with a section fully dedicated to this subject.

2.6 Representation

Time series tend to be large enough not to fit into main memory (Lin et al., 2003). This is a problem because disk accesses are orders of magnitude slower than main memory accesses. For that reason, a simple generic framework for time series data mining has been proposed (Faloutsos et al., 1994):

- Create an approximation of the data sufficiently small to fit in main memory, but sufficiently large not to lose the basic characteristics of the data;

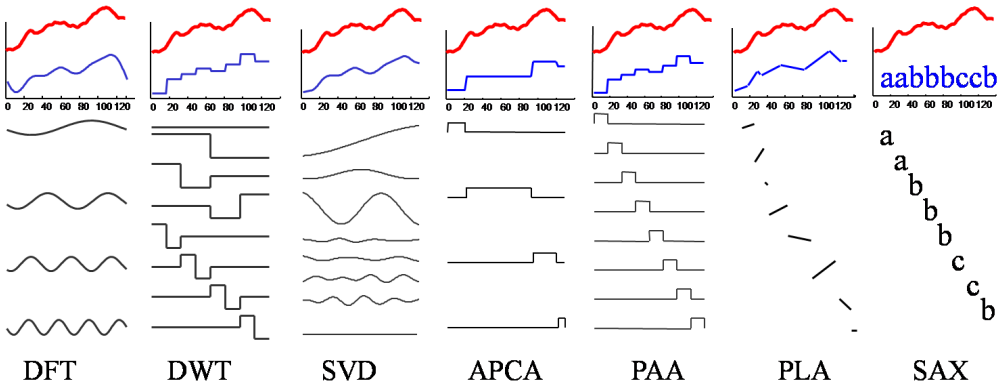


Figure 2.9: Most common time series representations available in the literature (from Keogh (2006))

- Produce a solution using the approximate representation;
- Verify the results with very few accesses to the original data in disk.

The quality of the approximation is thus a very important factor (Lin et al., 2003). If the approximation represents well the data, then the solution obtained from it should be similar (or ideally the same) to the solution using the raw time series. For that reason, since Faloutsos et al. (1994) a lot of interest has been shown by researchers in approximate representations of time series. Among the most important representations for time series are the Discrete Fourier Transform (DFT) (Faloutsos et al., 1994), Discrete Wavelet Transform (DWT) (Chan and Fu, 1999), Singular Value Decomposition (SVD) (Korn et al., 1997), Piecewise Models (Chakrabarti et al., 2002), and symbolic approximations (Lin et al., 2003). Figure 2.9 shows the most common time series representations. They can be categorized in Data Adaptive or non-Data Adaptive, as depicted in figure 2.10, whether they select the coefficients to use independently of the data, or not.

Among the non-Data Adaptive are the Spectral and Wavelets representations. The DFT is one example of Spectral representations, since it represents the frequency spectrum of the time series in terms of coefficients - ”pure” sine and cosine waves. By calculating the DFT of a time series, we are solely changing the representation, and not reducing the dimensionality of the series (Keogh, 2006). In order to take one step ahead, we need to take the assumption that the first coefficients carry

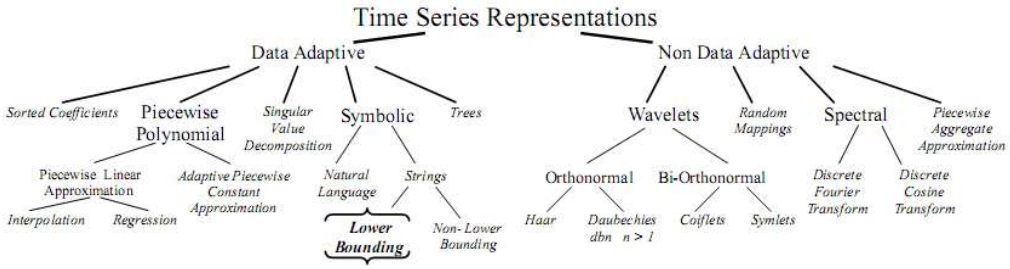


Figure 2.10: Hierarchy of time series representations (from Lin et al. (2003))

the most amount of information that describes the data. Using this observation, we truncate the coefficients by taking only the lower frequencies (up to a certain m), assuming the remainder are noise (Mörchen and Ultsch, 2007). Most of the time the assumption holds and the representation presents high compression rates, e.g. $n = 1000$, $m = 64$ with low reconstruction error. DFT have been successfully applied to most natural signals as implementations are widely available and typically provides good results. A similar representation to the DFT is the DWT. Instead of sin and cosine wave coefficients, the data is represented using a linear combination of Wavelet basis functions (Chan and Fu, 1999). Again, we are interested only in the first m coefficients. Typically the Haar Wavelet (Chan and Fu, 1999) is used. It is as powerful as others (Coiflets, Symlets and Daubechies) but easier to implement (Keogh, 2006). It has been widely used as a representation due to its good performance in compressing stationary signals. However, it presents better results if the time series size is a power of two. In non-Adaptive representations the first coefficients are selected, even if they are not the frequencies with the most power spectral density (coefficient selection is "blind"). Another non-Data Adaptive representation is the Piecewise Aggregate Approximation (Chakrabarti et al., 2002). In this representation the time series is divided into a number of segments or frames of fixed length. The segments are then represented by their average value. Despite its simplicity, this representation is widely used and is the heart of the Symbolic Aggregate Approximation (SAX)(Lin et al., 2003). We will cover this representation in detail in chapter 3.

Data Adaptive representations select the coefficients by taking information about the data into consideration. Both DFT and DWT may be used in this approach, by

selecting not the first coefficients but the coefficients carrying most energy (Mörchen and Ultsch, 2007). The SVD is similar to spectral and wavelets approaches as they represent the time series according to a set of coefficients representing shapes (Keogh, 2006) that are then truncated. This technique returns a set of descending coefficients that describe the variance of the data (thus it is data dependent) - eigenwaves (Chakrabarti et al., 2002). The SVD representation approximates the data in a faithful and meaningful way. However it is computationally expensive to calculate and can not be used in an incremental (online) fashion (Keogh, 2006) (see section 2.2). Piecewise Polynomial representations include Piecewise Linear Approximation (PLA) (Keogh and Pazzani, 1999) and Adaptive Piecewise Constant Approximation (APCA) (Chakrabarti et al., 2002). Both are similar to PAA because they divide the time series into frames. In the former, the frames have fixed size and each frame is represented by a straight line with a certain slope. The latter is a generalization of PAA where each frame has arbitrary length (Chakrabarti et al., 2002). Symbolic approximations represent the time series as a sequence of symbols. The main aspect in these representations is how one or more numerical values are mapped to a particular symbol. By using symbols one can re-use the wealth of already existing algorithms from string processing and bioinformatics (Lin et al., 2003).

The generic data mining framework is useless when the similarity results obtained with approximate representations of time series is completely different from the distance calculations that would have occurred using the original raw data. If we have the guarantee that the distance in the approximation space lower bounds the true distance in the raw time series, then we can develop our algorithms and operate them efficiently in the approximate reduced space. This fact was shown in Faloutsos et al. (1994). The intuition behind lower bounding the true distance can be explained by an example: if we have two long raw time series and compare them using the ED, obtain the DFT of both time series and compare the top coefficients with the ED, then we have the guarantee that the ED in the representation space is always lower or equal to the ED in the original space. Nearly all the representations in figure 2.9

have a lower bounding distance defined. This aspect needs to be addressed when introducing a novel representation. For example, in Lin et al. (2003) the authors introduced the new symbolic representation SAX, defined a distance measure for it, and then proved that such distance measure lower bounds the ED in the original space.

In order to choose one particular representation of time series we need to consider several characteristics (Lin et al., 2003), such as: computational complexity; indexing power; reconstruction error; compression rate; ability to support non-Euclidean distance measures and lower bounding distances; applicability in online time series; particular requirements (such as length); number of parameters necessary; ease of implementation; availability of algorithms for the representation; the application domain and task at hand.

2.7 Symbolic Aggregate Approximation

Time series approximate representations have been the subject of close inspection in the previous section. They are widely used in time series data mining in general, and in motif discovery in particular. They are the "heart" of most motif discovery algorithms available in the literature. Specifically, the Symbolic Aggregate Approximation (SAX) algorithm introduced in Lin et al. (2003), has been widely accepted as the representation to use in this task (Lin et al., 2007). This thesis is no exception. We use SAX extensively throughout our work. In this section we review and discuss the SAX time series representation.

As a symbolic approximation, SAX converts the original real time series T of length n into a sequence of w symbols – *word* – belonging to an alphabet of size $|\sum|$, or a. For example, $\sum = \{0, 1, 2, 3\}$. The alphabet size of a given SAX word is called *resolution*. This operation is represented by the following notation: $SAX(T, w, \sum)$. The name "aggregate" comes from the fact that it is based on the Piecewise Aggregate Approximation (PAA) (Keogh et al., 2001). This approximation is calculated

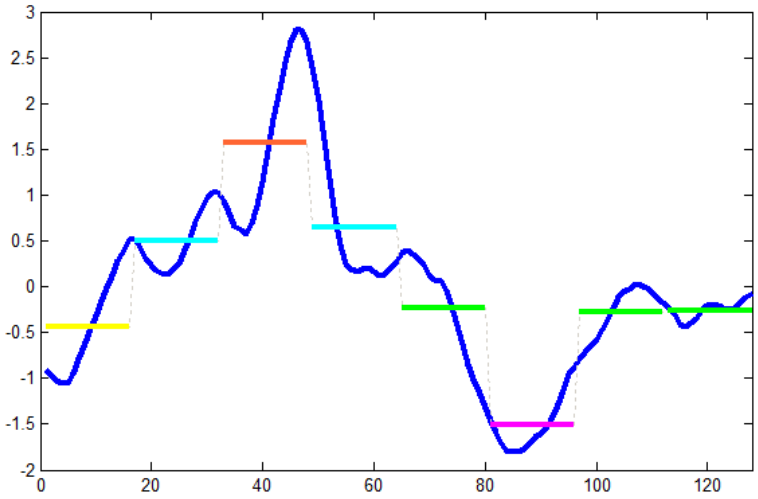


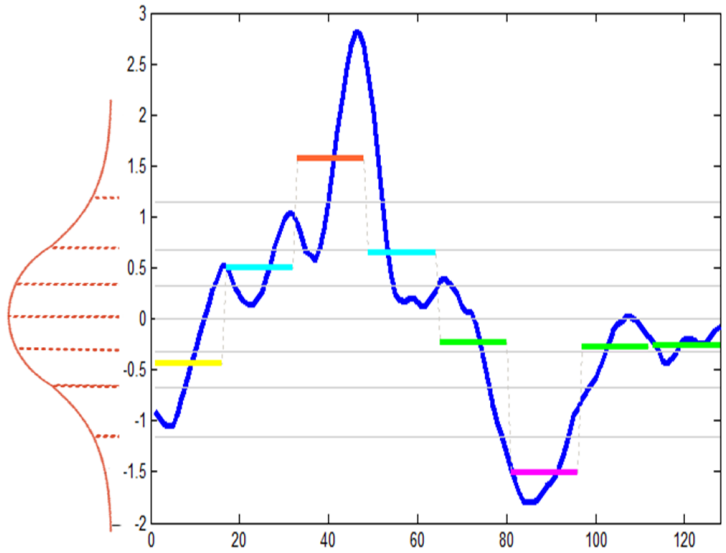
Figure 2.11: PAA representation with $n = 128$, $w = 8$ and 16 data points per symbol.

in a succession of steps:

- First, the dimensionality (length) of the time series is reduced by dividing it into w segments with the same length ($\frac{n}{w}$), by using the PAA algorithm. This algorithm calculates the average of all the values in each segment and then represents each segment by the calculated average. Figure 2.11 shows this procedure.

- Then, it is necessary to divide the amplitude of the time series into intervals, so that one symbol can be attributed to each interval. There are many techniques to perform this division, such as using equal sized intervals. However, this technique does not provide good results, because there are symbols that appear more frequently than others, as the distribution is not uniform. It is desirable to have a technique to obtain equiprobable intervals. As the sequence is normalized (section 2.5) it can be assumed that the sequence follows a Gaussian distribution. Therefore $|\sum|$ equiprobable intervals are obtained by using breakpoints that produce the same area under the Gaussian curve, as shown in figure 2.12, where $a = 8$ is used. These breakpoints can be obtained using a statistical table, for the desired alphabet size (a).

- Finally, symbols are obtained from the intervals defined by breakpoints. The



a / β_i	2	4	8	16	32
β_1	0.00	-0.67	-1.15	-1.53	-1.87
β_2		0.00	-0.67	-1.15	-1.54
β_3		0.67	-0.32	-0.89	-1.32
β_4			0.00	-0.67	-1.16
β_5			0.32	-0.49	-1.01
β_6			0.67	-0.32	-0.89
β_7			1.15	-0.16	-0.78
β_8				0.00	-0.68
β_9				0.16	-0.58
...				...	...
β_{14}				1.15	-0.16
β_{15}				1.53	-0.08
β_{16}					0.00
β_{17}					0.07
β_{18}					0.15
...					...
β_{31}					1.86

Figure 2.12: Intervals obtained by using equiprobable intervals according to the Gaussian distribution and $a = 8$

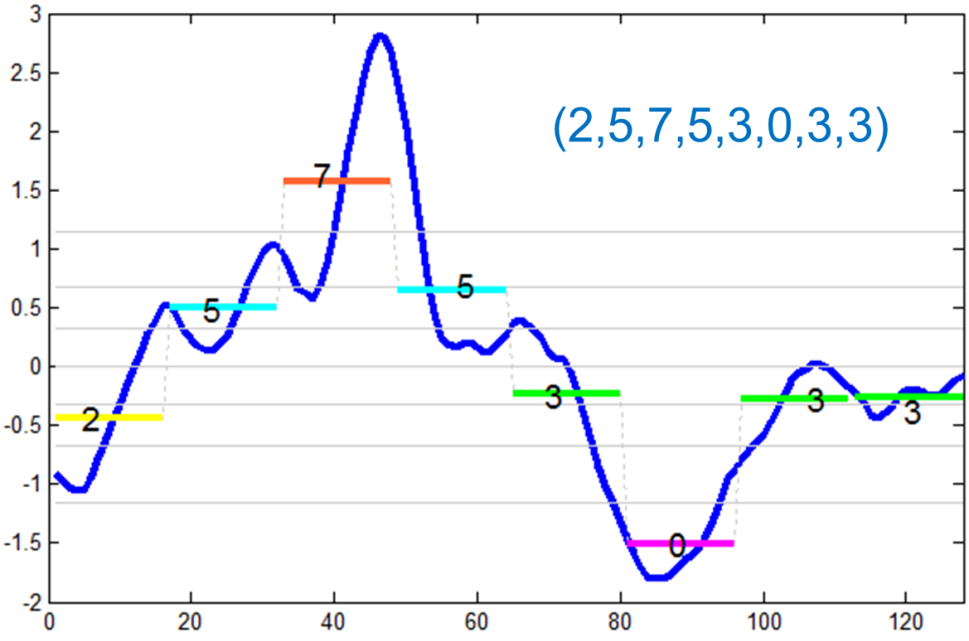


Figure 2.13: Attribution of the symbols to the PAA segments according to equiprobable intervals

segments below the smallest breakpoint are attributed the 0 symbol, the segments between the first and second breakpoints the symbol 1, and so forth. Figure 2.13 depicts this idea.

Having reduced the dimensionality of the original time series to w symbols, a distance measure for this representation is required. As discussed in the previous section, we need a distance measure that is correlated to the true distance that would have been obtained in the original time series. This can be guaranteed by having a distance in the representation that lower bounds the true distance. With that in mind, the authors define a distance measure between approximations $\hat{Q}$ and $\hat{C}$ of time series Q and C, similar to the Euclidean Distance, as follows:

$$\text{MINDIST}(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (dist(\hat{q}_i, \hat{c}_i))^2} \tag{2.3}$$

The $dist()$ function is a string distance function and is calculated according to expression (6) in the original paper. This function can be pre-calculated and then consulted for the given alphabet size.

The SAX representation has been widely used in the time series data mining community. The most important features of this algorithm is that is reduces the dimensionality and lower bounds the true distance of the original time series. Despite losing some of the information in the reduction process, it conserves the overall shape of the time series. The average calculation in the PAA technique is robust to noise, as it smooths the time series. However, if the sudden variations are the main aspect of the application domain, the smooth process caused by averaging can be a problem. A large number of algorithms from the biological and text mining communities is available to mine symbolic sequences (see section 3.5.1.3 and chapter 5). Finally, it can be used in streaming algorithms, which are present in a wide range of domains.

However, SAX also has some limitations. It has several parameters which may not be trivial to tune. These are the alphabet length $\sum$, and the number of segments w. Although the original paper presents some empirical evidence that the parameters are not too critical, the experience with our implementation of SAX suggests that they should be tuned with caution. If they are not set to their optimal value (or close), they can return very unintuitive results and the time series can lose its overall

shape. Besides that, it assumes that time series follow a Gaussian distribution, but there are domains in which this is not true (Mörchen and Ultsch, 2007). The number of segments w must be divisible by the data length n, which can be a problem has time series have arbitrary length that may not have divisors, for example. Typically the time series is re-sampled to a number that is divisible by 10, or by a power of two. However, the authors have recently implemented a version of SAX which seems to overcome this limitation. We incorporate this implementation in the algorithms presented in this thesis. The size of the alphabetic is also currently limited[*1] to 20. Other limitation is that after the approximation there is no reference to time. We only retain an ordered sequence of discrete symbols. The size of the segments is also fixed to $\frac{n}{w}$, whereas one would expect to have different granularities in the segment size or an adaptive segment length. It is also desirable to be able to work with different alphabet sizes, even within the same SAX word. The missing data is assumed to be dealt before the actual approximation transformation begins. Certain types of data where spikes are important to preserve (are not noise) may lose information due to the effect of averaging the data points in each segment.

Whereas there have been many research proposals that use SAX in some part of their work, there has been few research in extending the original approach (Lin et al., 2007). Some of these extensions add extra information to each symbol, such as the minimum and maximum of each segment (Lkhagva et al., 2006); and allow the multiresolution of both segment lengths and alphabet sizes (Lin et al., 2007). Future work aims to overcome some of SAX limitations and supporting more clever distances measures such as DTW (Lin et al., 2007).

Recently, the "classic" SAX has been extended by the original authors. The novel iSAX (Shieh and Keogh, 2008) has been introduced as a generalization of SAX for indexing and mining terabyte sized datasets. The iSAX representation extends classic SAX by allowing different alphabet sizes, or *resolutions*, for the same word. To avoid ambiguity, the resolution of each symbol needs to be made clear in the iSAX word. For example, $w = \{2^4, 0^8, 7^8, 9^{16}\}$. It is this enhancement that enables

[*1] In fact, we have extended the alphabet size to 512.

the creation of a time series index. However, for the scope of this thesis, we are not interested in the indexing capabilities of iSAX. In order to assist calculations, binary numbers are used instead of actual symbols. For clarity, these are typically displayed in their decimal format. However, it is indifferent to use integers starting at 0, arabic alphabet starting at a, or binary digits for other reason besides interpretability.

Converting an iSAX word from a higher to a lower resolution is simple: one just needs to ignore one trailing bit as we reduce the resolution by half. However the opposite is not true, since one can have several possibilities for the higher resolution. We need to convert the original time series to the new resolution if we want the correct result. Details on how this step is performed in an efficient way will be described in chapter 4. Figure 2.14 shows the same time series converted to resolutions 4 and 16. Notice that as the resolution increases, the intervals narrow considerably.

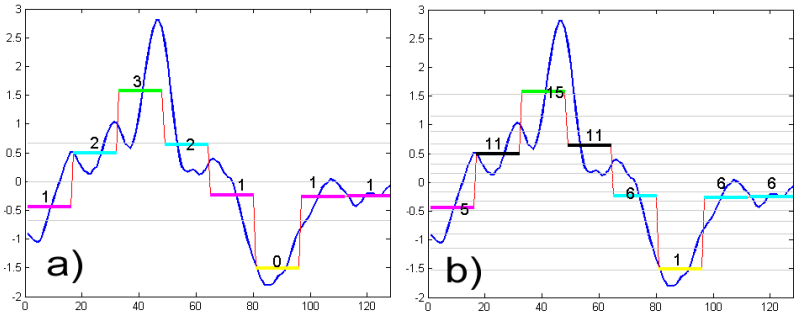


Figure 2.14: Example of the SAX conversion process for a time series with length 128, $w = 8$ and resolutions: a) 4, b) 16. Image generated by MATLAB and code provided by SAX authors Lin et al. (2003)

In fact, the algorithm proposed in chapter 4 is based on iSAX's ability of interleaving between the several resolutions.

In chapter 6 we present an automatic approach to calculate the SAX's parameters automatically.

2.8 Parameters

Most data mining algorithms are parametrizable in some way. Although parameters enable an algorithm to be applied to a wider domain of applications, when they

are unintuitive to set they can become a problem. Examples of these type of parameters are "size of matrix X", "sliding window length", or "minimum information gain threshold". As the right value to set the parameters is not clear even to the domain expert, the only solution is to probe the best parameter configuration. This is only possible for small size datasets. However, most of real-world datasets are of moderate or large size, while some are massive (Terabyte sized datasets). Take for example a time series database with one terabyte. To execute a naive motif discovery algorithm in this database can take one day. To test for the best parameter configuration with 3 parameters would take about a year, assuming only 8 possible value per parameter. In practice, the number of parameters and their ranges could easily surpass these values.

For that reason, unintuitive parameters should be avoided in data mining in general and in time series data mining in particular. Besides that, algorithms that present many intuitive parameters should also be avoided. The reasons are: they are typically difficult to implement, which makes them hard to reproduce; they tend to make the significance of a contribution hard to evaluate; they often cause the algorithm to over-fit to the training data and hard to generalize to unseen data; and make the algorithm useless to the common user, as he can not properly determine the best parameter configuration (Keogh et al., 2007).

Despite the fact that parameters should generally be avoided, sometimes they are an absolute necessity. In those cases, automatic strategies to tune the parameters to their best setting are necessary and desirable. Instead of using an iterative method – testing the values of a parameter experimentally by trying all or some possible values, they should take one step ahead and use an unsupervised method to efficiently derive the best parameters. One example of such technique is used in Yankov et al. (2007a) to properly tune the parameters of their motif discovery algorithm, or using an analytica approach, as in section 6.

- 3 -

State of the Art in Time Series Motif Discovery

3.1 Introduction

Time series are ubiquitous. They are produced in virtually every field in large amounts. They meet the inherant human desire to record observations over time. For example, one hour of ECG time series typically generates 1 gigabyte of data (Keogh, 2006). It is highly desirable to extract previously unknown and potencially useful knowledge from these data, i.e. mine the data, to obtain useful information such as: finding a pattern in ECG that precedes a heart attack 50% of the times. As we have seen, several types of tasks have been performed in time series data mining: classification, clustering, anomaly detection, query by content, motif discovery and visualization. Motif discovery is a particularly challenging task as it aims to find previously unknown recurrent patterns from time series data. In this sense, it walks toward the definition of data mining – to search for non-trivial, previously unknown implicit and potentially useful information from the data. So to say, we are letting the data "speak" to us, because we are not setting any constraint in the knowledge to be learnt. This is in contrast to supervised algorithms such as those used in classification, where a training set is used to learn a model that fits the data, eventually introducing a bias towards the instances in that learning dataset. Another important advantage of using unsupervised algorithms is that no training phase is required (Vahdatpour et al., 2009). In motif discovery the patterns emerge

as a natural consequence of the mining process, provided care is taken in choosing the parameters of a particular algorithm.

Surprisingly, much research is still lacking on learning hierarchy, structure, and patterns from time series (Li and Lin, 2010). These patterns, also known as *motifs*, provide useful information to the domain experts about their application domain (Ferreira et al., 2006) and can summarize the time series database. Extracting motifs can provide valuable insights on hidden semantics of the data and their underlying process and can potentially be used to ease other data mining tasks as classification and forecasting (Li and Lin, 2010).

For that reason, motif discovery has been used in areas as different as medicine, telecommunications, web, motion-capture and sensor networks. Fig. 3.1 shows an example of a time series with 3 different motifs (displayed in blue, green and red), as typically outputted by existing motif discovery algorithms.

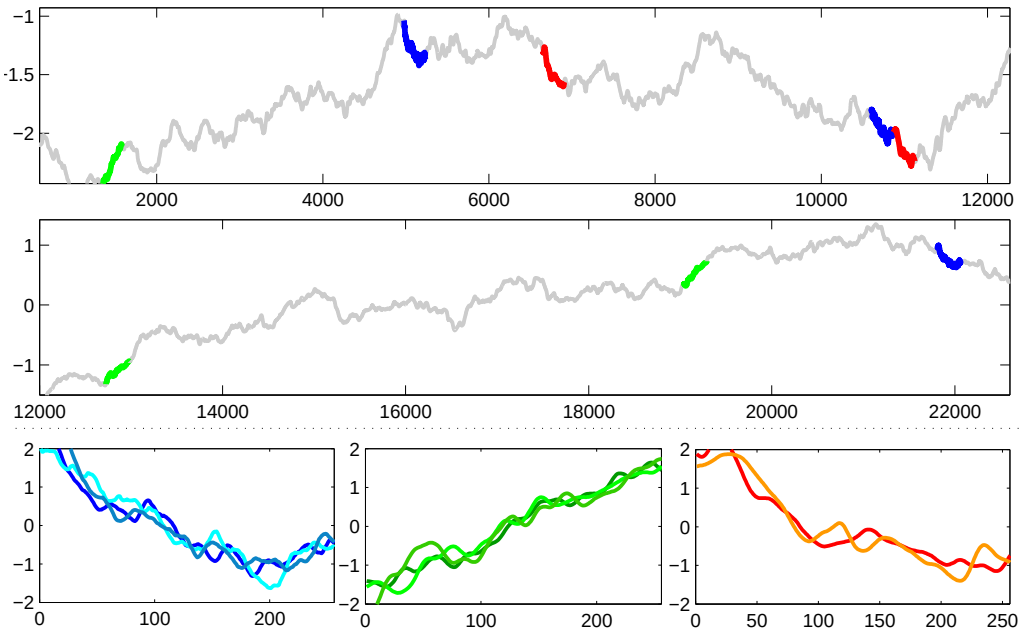


Figure 3.1: Example of a time series with several motifs. Above: in its original context; below: detail of each motif. Blue, Green: 3 instances; Red: 2 instances.

Since the problem formulation in Lin et al. (2002), a large number of proposals have been introduced (Castro and Azevedo, 2010; Chiu et al., 2003; Ferreira et al., 2006; Minnen et al., 2007a; Mueen and Keogh, 2010; Mueen et al., 2009a,b; Oates, 2002;

Tanaka et al., 2005; Yankov et al., 2007a). They differ in the way they approach the problem: whether they consider exact or approximate motifs, to find frequent or nearest-neighbor motifs (the chosen motif definition), to handle univariate or multivariate series, streaming or disk-resident data; and also on algorithmic details, as the space and time efficiency, time series representation techniques used, ability to handle motifs with different lengths, number of parameters, etc. Autonomously learning the motifs is difficult because their number, location, length, and shape are all unknown (Minnen et al., 2007c).

In this chapter we aim to provide a comprehensive review of the state of the art in time series motif discovery.

3.2 Definitions

Time series motifs have also been referred to in the literature as *recurring patterns* (Oates, 2002), *frequent trends* (Udechukwu et al., 2004), *approximately repeated sequences* (Chiu et al., 2003), *perceptual primitives* (Minnen et al., 2007c), *recurrent trends* (Jensen et al., 2005), *shapes* (Fuchs et al., 2009), *characteristic local patterns* (Buza and Schmidt-Thieme, 2010), or simply *temporal patterns*, *episodes*, and *frequent subsequences*.

There are two main categories of motif definitions in the literature. They are the *frequent motif* and the *nearest-neighbor* motif definitions.

3.2.1 Frequent motif

This category is based on the intuition that a motif is a previously unknown recurrent pattern in time series data. We now describe the most representative definition of motif in the literature that falls into this category.

The first work to propose the motif discovery problem (Lin et al., 2002) defined motif according to the concept of subsequence *match*. This concept is introduced

regarding a distance measure d, and the user-defined *range* (R) and *motif length* (m) parameters.

Definition 3.1. (Match) Given a positive real number R and a time series T containing two subsequences S_1 and S_2 of length m, if $d(S_1, S_2) \leq R$, then S_1 and S_2 match (Lin et al., 2002).

It is straightforward to observe that the subsequences with the best matches, i.e. the subsequences with the smallest distance between them, are subsequences that overlap. For example, the sequence starting at point 100 and ending at 200, and the ones starting at 101 up to 110 (ending at 101 up to 210). They share the majority of their elements and therefore are expected to have very small distance. These matches have little interest for the user, as they are meaningless. They are called trivial matches.

We now formally define this concept:

Definition 3.2. (Trivial match) Given a time series T, containing a subsequence S_1 beginning at position p and a matching subsequence S_2 beginning at q, we say that S_1 is a trivial match to S_2 if either $p = q$ or there does not exist a subsequence S' beginning at q' such that $d(S_1, S') > R$, and either $q < q' < p$ or $p < q' < q$ (Lin et al., 2002).

According to Lin et al. (2002), the concept of motif is formalized regarding the number of matches of each subsequence.

Definition 3.3. (K-Motif) Given a time series T, a subsequence length m and a range R, the K^{th} most significant motif in T is the subsequence S_k that has the K^{th} highest count of non-trivial matches, and satisfies $d(S_k, S_i) > 2R$, for all $1 \leq i < K$.

The expression $D(S_k, S_i) > 2R$ enforces the elements in different motifs to be mutually exclusive.

This motif definition comes towards the intuition of motif we aim to find: a previously unknown recurrent pattern in time series. It requires two parameters: the

motif range R – minimum distance for two subsequences to be considered a match, and the motif length m – the length of the subsequences. This definition has also been known as *range motif* due to the use of a range parameter. However, we instead call this category *frequency motif*. Our aim is to open it for other definitions that find frequent motifs while not using a range parameter.

3.2.2 Nearest-neighbor motif

The alternate definition in the literature has been proposed in Yankov et al. (2007a). In this work, a motif is defined in nearest-neighbor terms. The intuition is that the two subsequences that are nearest-neighbors (in terms of distance) in the time series represent the most significant motif, the second nearest-neighbors the second most significant, and so forth. This novel definition is proposed to alleviate the problem of setting the unintuitive R parameter of the previous definition, while claiming to be equivalent. The main difference is that this definition aims to discover subsequence pairs, i.e. the ones with the smallest distance, rather than the subsequences with the highest number of matches.

Similarly to the previous definition, we wish to avoid trivial neighbors.

Definition 3.4. (Non-trivial neighbor) Let S_i and S_j be two subsequences of length m in time series T of length n. We say that S_j, $1 \leq j \leq n - m + 1$ is a non-trivial neighbor of S_i, $1 \leq i \leq n - m + 1$, if there exists a subsequence S_{i_1} such that $i < i_1 < j$ and $d(S_i, S_{i_1}) > D(S_i, S_j)$.

Definition 3.5. (Non-trivial nearest-neighbor) The nearest non-trivial neighbor to S_i, is the non-trivial neighbor S_j with minimal distance $D(S_i, S_j)$

We are now ready to formalize the notion of nearest-neighbor motif.

Definition 3.6. (Nearest-Neighbor Motif) The most significant nearest-neighbor motif of length m in time series T of length n, is the subsequence S_i, $1 \leq i \leq n-m+1$ which has minimal distance to its non-trivial nearest-neighbor S_j, $1 \leq j \leq n-m+1$ (Yankov et al., 2007a).

Computing the subsequence with the largest number of neighbors is a different problem from calculating the two subsequences in the entire database with the smallest distance between them. For example, a pair of subsequences can be considered the most significant nearest-neighbor motif. However, in the same time series, it can be considered the least significant frequent motif. This can occur if these two subsequences are very close (distance-wise) to each other, but far from the remaining subsequences of the time series. Proximity does not imply frequency. It is straightforward to observe that frequency also does not imply close proximity. For example, the most frequent motif in the database with 100 matches, whose matches happen to be just below R, but there is no single top-10 nearest-neighbor among them. It has been claimed in Mueen et al. (2009a) that range motifs and other motif definitions can be calculated with "inconsequential overhead" using the nearest-neighbor as seed. It is also claimed that nearest-neighbor problem is the core of motif discovery. Given the counter-examples that we have presented, we disagree. We do not intend to argue that one definition is better than the other. In fact, both definitions can produce very interesting and meaningful patterns. Ultimately, they are two different problems and each should be applied according to the application and desirable type of pattern. When the goal is to find previously unknown recurrent patterns, the frequent motif definition should be used. When the user intends to find the closest pairs of patterns, the nearest-neighbor definition should be selected. The claim that nearest-neighbor motifs can be used as a seed to find frequent motifs is true. In fact, starting with a sorted list of neighbors (matches) is certainly advantageous. However, it is hard to calculate this list in less than quadratic time (as it will be shown throughout this survey). Computing exact frequent motifs can also be done in quadratic time, or approximately in linear time. We believe that the time necessary to calculate motif seeds does not justify using this technique when the goal is to compute motifs under other definitions.

3.2.3 Time series databases

Times series data mining algorithms usually extract the subsequences by sliding a window of a given length over the time series as shown in Fig. 2.6. We now formalize this concept.

Definition 3.7. (Sliding window) Given a time series T of length n, and a user-defined subsequence length of m, a matrix X of all possible subsequences can be built by sliding a window of size m across T and placing subsequence S_i in the i^{th} row of X. The size of matrix X is $(n - m + 1)$ by m (Chiu et al., 2003).

Typically, a step of one is used in the sliding window process. That is, the first subsequence to be extracted is $S_1 = (t_1, \ldots, t_{1+m})$, the second $S_2 = (t_2, \ldots, t_{2+m})$, and so forth. However, a step greater than 1 can be used. For example, with a step of 5, S_1 would start at t_1, and S_2 would start at t_6. In fact, a step greater than one can be used to avoid trivial matches (Castro and Azevedo, 2010).

In some applications, instead of a single long time series, we are interested in mining a collection of time series with arbitrary lengths.

Definition 3.8. (Time series database) A time series database D is a set of |D| unordered time series (Mueen et al., 2009b).

In practice, to slide a window through a long time series for the purpose of extracting subsequences is similar to handling each subsequence as a different time series. In the former, each row X_i in the X matrix contains subsequence S_i. In the latter, each time series D_i in the D database also contains S_i. It is straightforward to observe that the $n - m + 1$ by m matrix X is our database D after the sliding window process. However, it is necessary to account for trivial matches. This is typically achieved by taking a step greater than 1 in the sliding window process. As the database can contain time series with different sizes, it is often easier to control the overlap percentage between two consecutive windows.

For simplicity, we treat each subsequence of database as a different time series in D.

3.3 Dimensions

Motif discovery algorithms differ in the way they tackle the problem. They can be tailored to find frequent or nearest-neighbor motifs. Whether they consider exact or approximate motifs, handle multivariate or univariate time series, execute in streaming or batch mode, and find motifs with different or fixed length are other dimensions of the algorithms. Also, the time series representation used (approximate algorithms), similarity measure, their robustness to noise, space and time efficiency, and specific algorithmic details, such as the number of parameters. Finally, the peculiarities of the motifs: support, distance, cardinality, length, dimension, etc. (Mueen et al., 2009a; Wang et al., 2011). There are algorithms specifically tailored to find motifs in sets of time series (Futschik and Carlisle, 2005). As we have seen in the previous section, this distinction does not exist in practice. Therefore we exclude this dimension from consideration.

To summarize, the dimensions we consider to categorize TSDM algorithms are:

- Frequent or nearest-neighbor motif definition;
- Exact or approximate algorithm;
- Arbitrary or fixed length;
- Multivariate or univariate time series;
- SAX based / Random projection based (approximate algorithms);
- Streaming or batch execution;
- Time series representation (for approximate algorithms);
- Euclidean distance or Dynamic Time Warping;
- Robustness to noise;
- Time efficiency;

- Space efficiency;
- Amount of parameters.

Table 3.1 summarizes the analysis of existing algorithms in the above described characteristics.

Table 3.1: Existing motif discovery algorithms characteristics.

Dimensions	Frequent	Exact	Arbitrary length	Multivariate	SAX based	RP based	Robust to Noise	Streaming	Distance measure	Efficient	Memory efficient	Few parameters
Approaches												
Lin et al. (2002)	x	x							ED			
Oates (2002)	x	x	x	x			x	x	DTW			
Chiu et al. (2003)	x				x	x	x		ED			
Udechukwu et al. (2004)	x		x									
Tanaka et al. (2005)	x		x	x	x	x	x		DTW			
Ferreira et al. (2006)	x		x	x	x		x		Pearson			
Catalano et al. (2006)	x		x	x			x	x	DTW	x	x	
Jensen et al. (2005)	x		x						generic			
Minnen et al. (2007b)	x		x	x					DTW			x
Minnen et al. (2007a)	x			x	x	x	x		DTW			
Yankov et al. (2007a)					x	x	x		ED			x
Tang and Liao (2008)	x											
Fuchs et al. (2009)							x	x		x		
Wilson et al. (2008)	x		x									
Vahdatpour et al. (2009)	x		x	x	x	x	x					
Denton et al. (2009)												
Li and Lin (2010)	x		x									
Wang et al. (2010)	x			x								
Mohammad (2009)	x		x	x			x	x	DTW	x	x	
Mueen et al. (2009b)		x							ED	x		
Mueen et al. (2009a)		x							ED	x	x	
Mueen and Keogh (2010)		x						x	ED	x		
Lam et al. (2011)		x						x	ED	x	x	

3.4 Applications

In the last 10 years more than 200 papers have been published improving or just applying motif discovery. In this section we compile some of the applications where motifs have been used.

- Astrophysics (Chiu et al., 2003);
- Sensors (Castro and Azevedo, 2010; Hamid et al., 2005; Minnen et al., 2007b; Mohammad, 2009; Oates, 2002; Vahdatpour et al., 2009; Wang et al., 2010);
- Hydrology (Ouyang et al., 2010);
- Meteorology (McGovern et al., 2007);
- Medicine (Androulakis, 2005; Androulakis et al., 2005; Mueen et al., 2009a), dialysis treatment Knorr et al. (2009), surgical workflow analysis Ahmadi et al. (2009), epilepsy EEG (Yankov et al., 2007a), EEG (Mueen et al., 2009b), ECG (Li and Lin, 2010);
- Motion capture (Celly and Zordan, 2004; Hamid et al., 2005; Minnen et al., 2007b,c; Yankov et al., 2007a), sports motion capture (Tanaka et al., 2005);
- Shape recognition (Yankov et al., 2007a);
- Image recognition (Mueen et al., 2009a,b);
- Computer networking (Mueen et al., 2009a);
- Insect telemetry (Mueen et al., 2009b);
- Telecommunications (Castro and Azevedo, 2010);
- Robotics: (Mohammad, 2009; Mueen and Keogh, 2010; Oates, 2002);
- Language (Oates, 2002);
- Genetics (Androulakis, 2005; Androulakis et al., 2005; Jensen et al., 2005);

- Proteins (Castro and Azevedo, 2010; Ferreira et al., 2006; Jensen et al., 2005);
- Video motifs (Emonet et al., 2011);
- Music motifs (Muscariello et al., 2011; Wang et al., 2010);
- Finance (Wilson et al., 2008).

3.5 Motif Discovery Algorithms

In this section, we describe existing work in time series motif discovery. For simplicity, we divide them according to the motif definition involved and present them by chronological order.

3.5.1 Frequent Motifs

3.5.1.1 Lin et al. (2002)

The time series motif discovery problem was first proposed in 2002. In contrast to the usual task of finding known patterns in time series (query by content), the focus of the research is at finding previously unknown patterns in time series. Patterns are refer to time series subsequences. The concept of time series match is defined regarding a parameter R (range). Two subsequences match each other if the Euclidean Distance (ED) between them is lower than R. One important concept is that of a trivial match. It is believed that the best matches of almost all subsequences of a time series are the subsequences that immediatly precede or succeed them by one or two data points. Caution is necessary not to count these subsequences as frequent patterns in the motif discovery process. The concept of motif is generalized to $K - Motifs$, where the top K motifs are returned. The 1-Motif, or the most significant motif in the time series, is the subsequence that has most non-trivial subsequence matches. The concept of K-motifs is defined according to the R (range) and m (subsequence length) parameters.

The brute-force algorithm for calculating the 1-Motif of a given time series has quadratic complexity. Optimizations that can be derived from the symmetry property of the ED which allow to reduce the computation time in about one half. However, these optimizations are not space efficient (they require a large amount of storage). For that reason the authors introduce a sub-quadratic algorithm called Enumeration of Motifs through Matrix Approximation (EMMA). The basic idea is to use the triangular inequality property of the ED (section 2.4). To take full advantage of this property it is necessary to store a $m \times m$ matrix in main memory. This is infeasible even for moderately sized datasets. EMMA uses the SAX representation and the lower bounding property of its distance to create a hash function. This function places similar subsequences approximately together (with high probability) in small matrixes that can then be efficiently searched using the Approximate Distance Map (ADM) algorithm. The main optimizations lie on the use of smaller matrixes, and the ADM algorithm to skip a large number of calculations.

This algorithm is the seminal work on motif discovery in time series but presents some limitations. First, the sub-quadratic complexity is not sufficient in order to handle large datasets, as it is common in data mining. Next, unintuitive parameters need to be set, which is a difficult task even for a domain expert and can only be accomplished by testing different configurations.

3.5.1.2 Oates (2002)

Oates introduced an algorithm called PERUSE to find recurring patterns in multivariate time series in 2002. The algorithm can handle data sampled at different rates and motifs having arbitrary lengths. It allows for non-linear time warping as the model can be used on non-uniformly sampled data (Minnen, 2008). It uses the raw time series instead of discrete approximations. It also makes use of a sliding window to scan the entire time series. An exhaustive dynamic programming approach is used to discover the motifs. Namely, an HMM model with a Gaussian distribution is used. A hidden multinomial distribution is placed over all data frames that specify the

probability that the current motifs ends at that frame. Later, the multinomial and the motif's parameters are estimated iteratively using Expectation-Maximization (EM) (Minnen, 2008).

This algorithm presents a large computational complexity: the initialization procedure and the dynamic programming approach are expensive and the hidden multinomial needs to be re-estimated at all frames in every iterations (Minnen, 2008). Also, some stability problems can occur on the estimated models (Minnen et al., 2007a), as it assumes motifs are densely distributed – dense motif structure. Although this is the case in many datasets, in some others it may not hold.

3.5.1.3 Chiu et al. (2003)

This work introduces a probabilistic algorithm for the same task. Based on research in discrete motif discovery from the bioinformatics area (Buhler and Tompa, 2002), the authors develop an algorithm to find motifs more efficiently – Random Projection (RP). The algorithm is robust against noise and finds all motifs with high probability. As this algorithm is used extensively and is the basis of many other algorithms in the literature, we thoroughly describe it. It can be divided in several steps:

1. As the algorithm is inspired in work with biosequences (discrete symbols), the numeric time series is transformed in a discrete representation. The SAX time series representation described in 2.7 is used.

2. Each subsequence is taken from the original series using a sliding window and converted to its SAX representation.

3. The algorithm takes each discrete subsequence of length w (user-defined parameter, e.g. $w = 4$), and place it in a different row of matrix $\hat{S}$. Each row number points back to the original position in the symbolic time series. Matrix $\hat{S}$ has $k = n - m + 1$ rows and w columns (see figure 3.2).

4. Then, random projection begins: b columns (e.g. $b = 2$) are randomly selected from matrix $\hat{S}$ (Fig. 3.2 – middle).

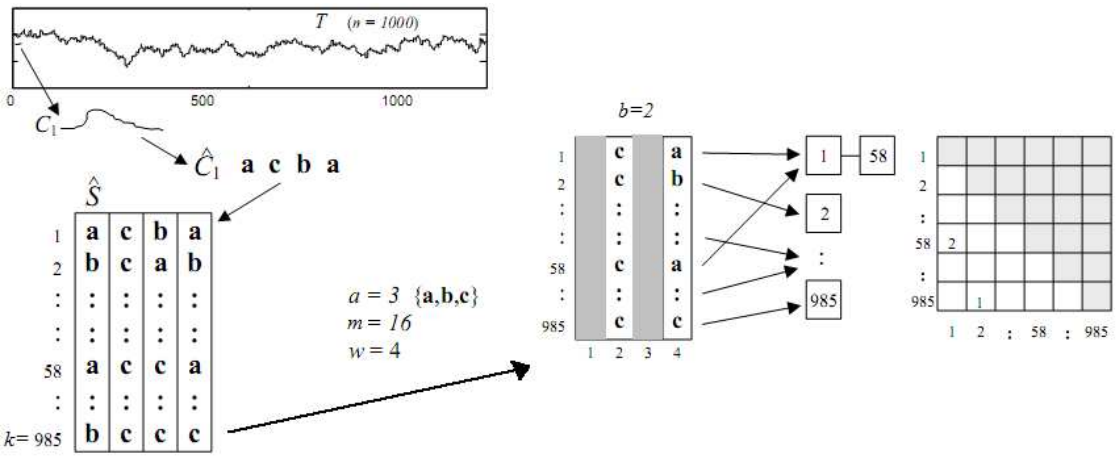


Figure 3.2: Random projection representation example (from Chiu et al. (2003))

5. By looking only at the symbols at the selected columns, it iterates through the k words of length b and count repetitions, storing the count in a $k \times k$ collision matrix (Fig. 3.2 – right)

6. The positions with the highest count in the collision matrix are regarded as candidate motifs.

7. The statistically significant entries in the collision matrix are verified if they are true motifs in the original subsequence. That is, the ED between the subsequences is calculated to see whether it is smaller than R. If yes, the subsequences are coined as true motifs.

8. The process either repeats for i iterations, stops at user request (anytime algorithm), or runs until statistical criteria are met. Typically, an iterative approach is selected.

This algorithm has been widely used in time series motif discovery since its introduction in 2003. Its strengths are its anytime behavior (providing good results fast), it claims linear complexity, low space complexity - the collision matrix is usually sparse and does not need to be stored on disk, and its robustness to noise or "don't care" regions. The fact that it is a probabilistic algorithm does not seem to affect the results, since only a few iterations are necessary to obtain all motifs with high probability. The longer the algorithm runs for, more accurate the results are.

Nevertheless, the algorithm also has some limitations. Namely, the several unintuitive parameters that need to be tuned. These are the range (R), motif length (m), number of columns in the $\hat{S}$ matrix, number of columns to use as a mask in random projection (b), and of course the SAX alphabet length ($\sum$) and number of segments (w). According to our experience, these parameters highly affect the outcome of the algorithm. For example, the R and m strongly depend on the characteristics of the time series at hand and need to be tuned in order to achieve good results. Otherwise, either no motifs, a massive number of motifs (wrong R), or unmeaninful motifs (wrong m) will be returned by the algorithm. Also, this algorithm claims it is "essentially" linear but this is simply not true. Each iteration of step 5 is quadratic in the length of the collision matrix. That is, k or $n - m + 1$, because it is necessary to perform two *for* loops to count every possible repetition. Even if we choose to perform an optimization to the algorithm: to first discretize the series and then take symbolic windows, it is still quadratic in $\frac{n}{s} - w + 1$. The denominator s is the number of time series points compressed to just one symbol: m/w, i.e. the numerosity reduction factor. For the example in Fig. 3.2, it is 4. Therefore it is expected that the algorithm is quadratic in approximately $n/4$, $n/8$, or $n/16$, depending on the selected SAX word length and motif length. The authors state that with certain optimizations it is possible that the collision matrix length reaches 22% of the original time series length. Even though, one iteration is still quadratic. The complexity is further increased because more than one single iteration is required. The authors state that a reasonable number of iterations is between 1 and 100. Therefore, to obtain the complexity of the algorithm the complexity of one iteration is multiplied by the expected number of iterations (average complexity), or the maximum number of iterations (worst case complexity).

Future work stated by the authors includes reducing the number of parameters, extending the algorithm to also find multivariate motifs, to create an online version of the algorithm, and support measures that take into account distortions in the time axis such as DTW.

3.5.1.4 Udechukwu et al. (2004)

This work proposed a linear time motif discovery algorithm. The time series is converted to symbolic form, a suffix tree is built and then the frequent patterns and trends are discovered. The proposed representation scheme is simple and uses the relative changes in the time series values to encode the series into a finite alphabet string. The scheme assigns each symbol according to the degree of steepness between every two neighbor time points. That is, the symbols are assigned according to angle between each time point and the next, ranging from $-90°$ to $90°$. The size of the alphabet can be user specified. This algorithm does not require the user to set a motif length. The algorithm is sufficient but is expected to be affected by noise, as many points in time series data are caused by noise. However, the authors state that the representation scheme can be used in a sample version of the series. That is, to consider only some points to calculate the angles, rather than all.

3.5.1.5 Tanaka et al. (2005)

An algorithm to find motifs in multivariate time series is proposed. The multivariate time series is projected into one univariate signal using Principal Component Analysis (where most information is retained). Then the Random Projection algorithm (Chiu et al., 2003) is used to handle the univariate case. The Minimum Description Length (MDL) principle is later applied to detect motifs of variable lengths, aiming to find the optimum length. This is achieved by using the SAX representation to discretize short windows and later assigning a different symbol to it. It remains to be investigated whether the possibly exponential number of unique symbols is manageable in practice (Minnen, 2008). As the number of unique symbols could grow exponentially regarding the length of the SAX string, further experimental analysis is necessary to ensure that the number of unique symbols is computationally feasible (Minnen, 2008). Motifs that occur on all dimensions are detected. However, the approach assumes the first principal component does capture enough information about the multivariate series. That is, the approach is limited to data that can be

correctly represented by the first principal component, as it is the only one used (Minnen et al., 2007d).

3.5.1.6 Ferreira et al. (2006)

In this work, the authors extend the definition of motifs to subsequences that may also present inverse behavior (shape). To account for this definition, the Pearson correlation coefficient is used. This measure returns a value between -1 (negatively correlated or inverse behavior) and $+1$ (maximally correlated). This measure exhibits the symmetry, positivity and triangular inequality properties. Therefore it can be considered a metric. The algorithm's parameters a minimum support δ, the motif length m, a minimum similarity R_{min} and a window frame length *deltaW* (to avoid trivial matches). It aims to find all the motifs according to the parameter's constraints, by using a three step approach. In the first step, the dimensionality (size) of the time series is reduced using the SAX technique. Second, all the subsequences of length m are compared to each other for similarity. Then, an agglomerative bottom-up approach begins: clusters of 2 subsequences are formed according to their similarity. These clusters are then extended to have arbitrary size. The triangular inequality property has an important role in the efficiency of this step. Trivial matches are also removed by taking into account the *deltaW* (distance between motifs) parameter. Finally, each retrieved motif length is extended until the similarity drops below the user-defined R_{min} threshold. A post-processing phase can also occur to further prune motifs based on information measures such as Information Gain.

The strengths of this algorithm are: it is able to handle multivariate time series, it presents an intuitive R parameter (e.g. $R = 1$ means maximal similarity), it allows to retrieve motifs of different sizes, inverse motifs are also retrieved, the use of the Person coefficient seems to capture well the notion of similarity, it retrieves all motifs under the parameter constraints (no false dismissals), and the intuition that cluster extension and motif length extension seem desirable in motif search. Its limitations

are the quadratic complexity, the need to load the entire dataset to main memory (which is unatenable even for moderately sized datasets), and also the need to tune four parameters (besides SAX's). Future work includes tackling the scalability of the algorithm, avoid putting the entire dataset in main memory (while maintaining the efficiency), and reducing the number of required parameters.

3.5.1.7 Catalano et al. (2006)

This work presents an efficient algorithm to find motifs in multivariate and streaming time series using random sampling. The algorithm is robust to noisy datasets. This algorithm executes in linear time and constant memory. To accomplish this, it assumes motifs are densely distributed. As it is based on random sampling and uses a small amount of memory, it can perform poorly when motifs are rare or occur sparsely, e.g. long series of human motion and physiological data (Mohammad, 2009). Typically, a larger amount of memory is necessary to detect a motif that only occurs either rarely or at widely spaced (Minnen, 2008). Also, it is necessary to set 6 parameters.

3.5.1.8 Jensen et al. (2005)

This work generalizes motif discovery over categorical sequences and time series data. It is composed of three steps: pairwise comparison of all subsequences of a given length, clustering of similar windows to form motifs, and motif concatenation to form maximal motifs. It is flexible as it allows arbitrary similarity metrics and clustering function. It finds all maximal motifs in terms of both length and composition.

However, it requires an exhaustive comparison among all subsequences of a particular length (quadratic complexity) – it does not scale. It also only supports variable length motifs by using a post-processing phase (Minnen, 2008). This can be a problem in time series data, as two very similar subsequences may appear quite different if only fixed-length segments are compared. Minnen (2008) shows the following interesting example: comparing ABCD and AABBCCDD with a window

length of four. Also, the paper claims that it is memory efficient, while stating that the peak memory usage for a sequence containing 65000 characters is 1GB. In time series data mining such dataset may be considered small. Such performance does not scale for medium or large data sets.

3.5.1.9 Minnen et al. (2007b)

This work views motif discovery as the problem of locating regions of high density in the space of all time series subsequences. Such regions capture the idea of a motif. Detecting a time series motif is very similar to computing the density around each subsequence, where the examples in the subsequence neighborhood are its non-trivial matches. The most significant motif is simply the sequence whose neighborhood has maximum density (i.e. largest number of neighbors) (Yankov et al., 2007a). This algorithm locates a set of candidate motif seeds by identifying subsequences located near high density regions. Then, it detects motif occurrences from the candidate set using a form of greedy mixture learning that allows variable-length occurrences. Namely, it uses HMM to fit the time series and likelihood scores to rank the candidates (Wang et al., 2010).

The approach reduces the number of user-specified parameters. However, it requires high computation (Wang et al., 2010).

3.5.1.10 Minnen et al. (2007c)

This approach proposes a method to automatically estimate the neighborhood size of each motif. This is useful because it is no longer necessary to specify a range-like parameter, which is crucial to mine correct and highly domain dependent motifs. Also, it permits the neighborhood size to vary among the different motifs. This is very hard to define manually and *a priori*. To detect the appropriate range it finds the knee in the distances between motif seeds and other motif candidates curve (Mohammad, 2009). The authors also propose an approach to automatically detect the motif length (Vahdatpour et al., 2009).

3.5.1.11 Minnen et al. (2007a)

This paper introduces an algorithm that finds motifs in multivariate time series in linear time. The algorithm generalizes previous work of the same authors (Minnen et al., 2007c), where patterns had to span across all time series. In this work, motifs may span only across some of the time series or dimensions. The algorithm is based on Random Projection (Chiu et al., 2003). The difference is due to the multivariate case, where a combined projection among the different time series is used. Whereas in Minnen et al. (2007c) the collision matrix is incremented only if all dimensions matched a particular subsequence, the collision matrix is now incremented once there is a match in one dimension. The DTW is also tested as the distance measure, but the complexity goes quadratic. The authors use the 10% Sakoe-Chiba band constraint in the warping path. However, it has been shown that using the LB_Keogh lower bounding technique makes DTW approximately linear (Ratanamahatana and Keogh, 2005).

The strengths of this algorithm are inherited from Random Projection, plus the ability to handle the multivariate case. Also, it creates the conditions to avoid Random Projection from growing quadratically. That is, it makes the distribution of the collisions sufficiently wide by dynamically adjusting the SAX's alphabet size and c parameters.

However, this approach assumes that motifs along all the dimensions are occurring synchronously, which restricts the applicability of the method to data coming from systems with highly accurate actions (e.g. space shuttles with accurate and almost noise free sensors (Vahdatpour et al., 2009).

3.5.1.12 Tang and Liao (2008)

This approach aims to improve random projection by providing a way to handle arbitrary lengths. The strategy is to start with a small m value and then concatenate the shorter patterns to discover the whole pattern. The paper states that one of

the drawbacks of the random projection is that it only identifies pairs of matches, rather than all occurrences of a pattern. This proposal overcomes this limitation by summarizing an average pattern based on the occurrences of a pattern.

3.5.1.13 Fuchs et al. (2009)

This work presents SwiftMotif, a technique for motif detection in time series. It supports data streams, noise, and allows local variability of time domain (uniform scaling). It is based on a fusion of probabilistic modelling with extremely fast polynomial least-squares approximation techniques. This leads to a new kind of motif representation and a new kind of (dis-)similarity (distance) measure for motifs. The former is basically a normal distribution with time-dependent, polynomial-shaped mean and constant variance. The latter is based on a divergence measure for probability densities. The contributions are as follows: a new representation of time series or time series segments which considers the uncertainty associated with the occurrence of certain motif candidates or motifs. Also, a novel measure for time series dissimilarity which is based on the representations and considers the local variability of time series. Finally, an algorithm that allows to segment time series, to model the time series within the segments, and to find similar or dissimilar segments online. The run time of this technique is extremely low which is due to the fact that the overall complexity of the technique is $O(N \cdot K)$ with N being the number of samples and K being the highest involved polynomial degree.

Recently, Mueen and Keogh (2010) stated that this work actually finds approximate motifs offline. Rather than discovering time series motifs online, it approximately filters them offline.

3.5.1.14 Wilson et al. (2008)

The paper introduces the motif tracking algorithm (MTA), a novel immune inspired pattern identification tool that is able to identify unknown motifs of unspecified length. The power of the algorithm comes from the fact that it uses a small number

of parameters with minimal assumptions regarding the data being examined or the underlying motifs. The algorithm identifies the presence of a motif population in a fast and efficient manner due to the utilization of an intuitive symbolic representation (SAX). It uses a genetic algorithm by first converting the time series to SAX and then mutating the patterns. The time series differences are used rather than the raw time series points. It is based on the range parameter and has quadratic complexity.

3.5.1.15 Vahdatpour et al. (2009)

The approach leverages the benefits of Random Projection (Chiu et al., 2003) and extends it by designing a clustering algorithm that constructs activity perceptual primitives from time series motifs. The major contribution of the method is its ability to discover multi dimensional motifs that have time and value irregularities, which is a common case in activity monitoring applications. The clustering is done on the coincidence graph which is based on the temporal coincidence of motifs in different time series dimensions. It also presents a feedback approach to improve the accuracy of single dimensional motif discovery process. The approach can discover motifs that are not synchronous across all time dimensions, having different length and timing characteristics.

3.5.1.16 Denton et al. (2009)

This paper introduces a clustering algorithm, that creates clusters exclusively from those subsequences that occur more frequently in a dataset than would be expected by random chance. As such, it partially adopts a pattern mining perspective into clustering. When subsequences are being labeled based on such clusters, they may remain without label. In fact, if the clustering was done on an unrelated time series it is expected that the subsequences should not receive a label. It shows that pattern-based clusters are indeed specific to the dataset for 7 out of 10 real-world sets tested, and for window-lengths up to 128 time points. While kernel-density-based clustering can be used to find clusters with similar properties for window sizes of 8-16 time

points, its performance degrades fast for increasing window sizes. The complete algorithm is summarized as follows: for each data point it examines the number of neighbors as a function of Euclidean distance from that point. This function (radial distribution) can be analytically calculated for a Gaussian random walk model, and a normalization that places data on a hypersphere. The number of observed data points, compared with the number of expected ones, is used as an indicator of how exceptional a data point is. The resulting cluster centers are altered such that each center is represented by the cluster with highest density within its range. After that, clusters are combined into groups, each of which receives a label.

3.5.1.17 Li and Lin (2010)

This work proposes a methodology to find approximate variable-length time series motif using an efficient grammar-based compression algorithm. Several algorithms were proposed to discover motifs of variable lengths. However, they either do so via post-processing, scale poorly, or discretize the whole data rather than considering overlapping subsequences, resulting in inaccurate and incomplete patterns found. The algorithm mitigates these shortcomings. In addition, it offers the advantage of discovering hierarchical structure, regularity and grammar from the data. Even though there are some known issues with the algorithm, the preliminary results are promising. They show that the grammar-based approach is able to find some important motifs and suggest that the new direction of using grammar-based algorithms for time series pattern discovery might be worth exploring.

3.5.1.18 Wang et al. (2010)

The paper examines an unsupervised search method to discover motifs from multivariate time series data. The method first scans the entire series to construct a list of candidate motifs in linear time. The list is then used to populate a sparse self-similarity matrix for further processing to generate the final selections. The proposed algorithm is efficient in both running time and memory storage.

3.5.1.19 Mohammad (2009)

This work extends Catalano et al. (2006) by studying the constrained motif discovery problem which provide a way to incorporate prior knowledge into the motif discovery process. Most unconstrained motif discovery problems can be transformed into constrained ones and provide two algorithms to solve such problem (Esling, P. and Agon, C., 2011) They try to find motifs utilizing information about their probable locations in the data series. This leads to a linear time algorithm with constant space complexity. The authors also show how to effectively convert unconstrained motif discovery problems into constrained motif discovery.

3.5.2 Nearest-neighbor Motifs

3.5.2.1 Yankov et al. (2007a)

This work is based on the intuition that even small stretching in the time axis of time series can dominate distance measures. For example, if the sampling rate of two time series is different the distance measures will present large values, even if they have the same shape. Motivated by this fact, this work extends the approach in Chiu et al. (2003) to allow the detection of motifs under uniform scaling. The definition of motif is also modified and it is no longer centred on an unintuitive range parameter. Motifs are now defined in nearest-neighbor terms: the most frequent motif of a time series is constituted by the subsequences that are the nearest to each other. In order to account for diffent scalings in the time axis, a uniform scaling distance metric is used instead of the Euclidean Distance. Basically, the idea is to re-scale the subsequences according to the best scaling factor. However, to obtain the best scaling factor we need to probe through all or most factors. The authors empirically demonstrate that only a few of the scaling factors need to be checked, because the others will probably present the same results. After re-scaling, the idea is to apply Random Projection. Another important contribution of this paper is that it defines an offline unsupervised manner to tune the parameters by using a small training set

from the domain at hand. The only parameter that needs to be actually defined by the user is the motif length.

The main contributions of this approach are: the new motif definition, the method to automatically tune most of the parameters and obviously the ability to handle different stretches of the time axis. Nearest neighbor motifs can be useful in some applications, as it is no longer necessary to include the unintuitive range parameter. However, the motif length parameter still has to be defined by the user. Further work includes extending the approach to multivariate time series, removing the motif length parameter and handle other distortions using for example DTW.

3.5.2.2 Mueen et al. (2009b)

The work proposes the first tractable exact motif discovery algorithm under the nearest-neighbor definition – MK. The algorithm is up to three orders of magnitude faster than brute-force. It is based on the notion of early abandoning the Euclidean Distance calculation when the current cumulative sum is greater than the best-so-far. The motif search is guided by heuristic information calculated by the linear ordering of the distance of an object with respect to a few random reference points. The observation is that if two subsequences are close in the original space they must also be close in the linear ordering (Li and Lin, 2010). MK is a sound contribution in the exact motif discovery search. However, the use of the Euclidean Distance directly in the raw data can give rise to robustness problems when dealing with noisy data. Euclidean distance can be highly affected by noise (Chiu et al., 2003). For example, by adding just 10% of the standard deviation of Gaussian noise, the top 10 motifs of the database change for most datasets. Also, when data does not fit in main memory it will perform a large number of random disk accesses, which may prevent the algorithm from scaling.

3.5.2.3 Mueen et al. (2009a)

This work describes a disk-aware algorithm to find exact time series motifs in multi-gigabyte databases, containing tens of millions of time series. It is based on a bottom-up search algorithm that simulates the merge steps of the divide-and-conquer approach. The main contribution is that the worst-case memory and I/O overheads are practical for implementation on very large-scale databases. The key difference with the optimal algorithm that makes it amenable for large databases is that the data is divided without reducing the number of dimensions and without changing the data order at any divide step. This allows to do a relatively small number of batched sequential accesses, rather than a huge number of random accesses. This can make about four orders of magnitude difference in the time it takes to find the motifs on disk-resident datasets.

3.5.2.4 Mueen and Keogh (2010)

This work proposes the first online motif discovery algorithm which monitors and maintains motifs exactly in real time over the most recent history of a stream. The algorithm has a worst-case update time which is linear to the window size and is extendible to maintain more complex pattern structures. In contrast, the current offline algorithms either need significant update time or require very costly pre-processing steps which online algorithms simply cannot afford. The key idea behind the algorithm is to use a nearest neighbors data structure to answer the motif query fast. Although this approach demonstrates a space-efficient algorithm for exact motif discovery in real time, in several real applications with large sliding window sizes it can be very space demanding. Furthermore, in many real applications, the discovery of top-k motifs may be more useful and meaningful than only the top-1 motif. It is non-trivial to modify the algorithm to the problem of online top-k motifs discovery. (Lam et al., 2011)

3.5.2.5 Lam et al. (2011)

This work generalizes the problem of online exact motif discovery to any k and propose space-efficient algorithms for solving it. The authors show that the algorithms are optimal in term of space. In the particular case when k = 1, they achieve better performance both in terms of space and time consumption than Mueen and Keogh (2010).

3.6 Motif based tasks

Motif discovery can be the basis of other data mining tasks, at it summarizes the time series database.

Hereby we compile a list of tasks motif discovery algorithms have been applied to.

- Classification (Buza and Schmidt-Thieme, 2010; Knorr et al., 2009);
- Rule Discovery (McGovern et al., 2007);
- Novelty detection (Rombo and Terracina, 2004);
- Clustering (Keogh and Lin, 2005; Rombo and Terracina, 2004);
- Summarization (Rombo and Terracina, 2004);
- Visualization (Lin et al., 2004).

- 4 -

Multiresolution Motif Discovery in Time Series

4.1 Introduction

The extraction of frequent patterns from a time series database is an important data mining task. These patterns, also known as motifs, provide useful insight to the domain expert about the problem at hand (Ferreira et al., 2006) and help summarize/represent the time series database. For that reason, motif discovery has been used in areas as different as telecommunications, medicine, web, motion-capture and sensor-networks. For example, in EEG time series (figure 4.1) a motif may be a pattern that usually precedes a seizure; in DNA it may be a sequence of symbols that has been preserved through evolution (Ferreira et al., 2006); in motion capture a particular gesture (e.g. throw ball); in telecommunications, a typical burst in traffic when major social events are located near an antenna. Figure 4.1 shows one example of such type of pattern in the context of EEG time series from Yankov et al. (2007a). This specific motif is detected in two different time series of the database. Figure 4.2 shows the three occurrences of the motif in figure 4.2 along the same axis after normalization. Normalization of the time series is required to remove offset and scaling effects (Lin et al., 2003). It has been shown that comparing time series that are not normalized is meaningless (Keogh and Kasetty, 2003). After normalization the series are very similar but not identical. The aim of this work is to find frequent patterns fast, but still be able to handle noise and other distortions

in the time series.

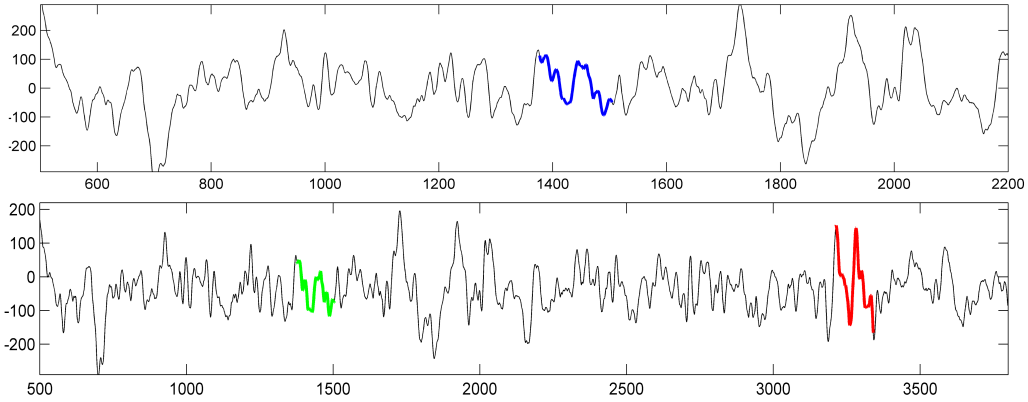


Figure 4.1: Example of a motif of length 128 in EEG time series in its original context.

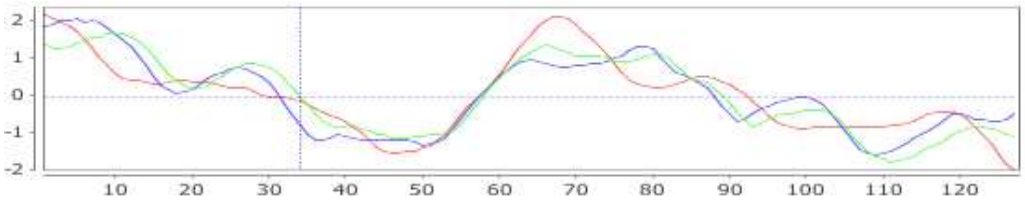


Figure 4.2: Example of the 3 instances of the motif in the same referential.

Most of the motif mining proposals heavily rely on random disk accesses to the disk database (Ferreira et al., 2006; Mueen et al., 2009b; Oates, 2002) which is inefficient. It is known in the database community that accessing 10% of a disk database randomly takes essentially the same time as traversing the entire database sequentially (Yankov et al., 2007b). Even for moderate size datasets this becomes an issue. Other techniques (Lin et al., 2002; Minnen et al., 2007a; Tanaka et al., 2005; Yankov et al., 2007a) tackle this problem by putting the entire dataset in main memory. The assumption that the data can fit in main memory is incorrect most of the time. While this may work in small synthetic datasets, it becomes an unfeasible task when using real world databases that typically present Gigabytes. Researchers have countered this problem by using approximate time series representations (Lin et al., 2003). Despite being less accurate than their exact counterparts, approximate algorithms present a relatively good trade-off between accuracy and efficiency. They are also typically robust to noise (Castro and Azevedo, 2010; Chiu et al., 2003). We believe approximate algorithms are the best option in many application domains due

to their time/space efficiency. For instance, sensor networks and telecommunication networks monitoring often require real time results. In these domains, the trade-off between execution time and accuracy of the solution clearly bends towards the former. The framework described in section 2.6 is typically used. The time series database is converted to a representation that requires less space but still retains most of the information in the series. Then, the converted dataset is loaded into main memory using one sequential disk scan. The problem is solved in main memory and only few accesses to the original disk data are required in order to confirm the results (Lin et al., 2003). It is straightforward to observe that loading the entire representation of the time series database in main memory is unfeasible for massive datasets. Also, the assumption that the representation of the time series fits in main memory may not hold in those cases, unless the dimensionality reduction factor yield by the new representation is large, causing the loss of the most important time series features. There are very specific applications where storing the representation is also not possible, such as sensor networks or mobile applications. These applications use devices where the amount of memory is limited, requiring space efficient algorithms. Furthermore, in some of the applications the entire dataset is not available and the algorithms are often not prepared to handle this important type of scenario e.g. data streams. A similar setting is that of massive databases. In these cases the amount of data on disk is very large and each element can only be visited once.

There are many methods designed for univariate time series. However, many time series are multivariate, for example multi-sensor systems (Minnen et al., 2007a). Most of the real world applications deal with several data sources and therefore, require methods to analyze multi dimensional data (Vahdatpour et al., 2009). Also, most existing approximate algorithms only find motifs at a single resolution i.e. using a fixed number of symbols in the time series representation. However, it desirable to handle different levels of the time series representation to achieve further insights about the data.

Data mining algorithms typically provide as output a set of patterns: the most frequent patterns, the nearest neighbor subsequences in the database, etc. We

believe that user interactivity is an important part of data mining. For that reason data mining and motif discovery algorithms should provide means to facilitate the development of user interactivity.

In this chapter, we tackle time series motif discovery as an approximate Top-K frequent pattern problem.

We achieve time efficiency by using a single sequential disk scan to read the time series database, a clever time series representation and a hashtable based counting technique. Memory efficiency is achieved by combining our method with the space-saving algorithm (Metwally et al., 2005), now applied to time series data mining. Our approach is based on the state of art time series representation – iSAX (Shieh and Keogh, 2008), leveraging its multiresolution property to derive motifs at different resolutions. Typically, an expensive motif verification step in the raw data is employed by approximate motif discovery algorithms. The motifs discovered in the approximate space are verified in the real data using a distance measure, e.g. Euclidean Distance (ED). We exploit the iSAX multiresolution capability to skip this expensive verification of motifs in the raw data. This property also enables the development of powerful visualization applications, allowing the navigation in the Top-K motifs hierarchy structure. This yields better understanding and intuition to the user about the database at hand. The single sequential disk pass makes the method a strong candidate for data streaming applications.

The remainder of the chapter is organized as follows: section 4.2 describes the multiresolution property, background and notation used throughout the chapter are described in section 4.3; section 4.4 introduces our approach; the experimental analysis is described in section 4.5; finally, we derive conclusions and describe future work.

4.2 Multiresolution

In this section we briefly describe the multiresolution property and its advantages.

The process of counting subsequences occurrences in time series is not trivial. The criteria for which one could consider one subsequence as the repetition of another are diverse. It is clear that counting only equal subsequences as repetitions is of no great use. One should use "similar" subsequences instead. Once this aspect is considered, the next question to address is what similarity definition one should use. To let the data mining practitioner set this as a parameter such as the range (R) is not an interesting approach, since this parameter is largely domain dependent and unintuitive to adjust. One could choose to select the pair of subsequences in the database that are the nearest to each other (*nearest-neighbor* motif definition). However, this solution does not take into account the frequency of the subsequences. We may have a pair of subsequences that are the nearest neighbors but are rare in the database (e.g. one occurrence). An interesting approach is to formulate the motif discovery problem as a top-K frequent pattern problem (Metwally et al., 2005). In this framework, we need a container where we can put similar time series with an adjustable margin of similarity. To be able to dynamically adjust the similarity degree is desirable in motif discovery. Figure 4.3 highlights this idea.

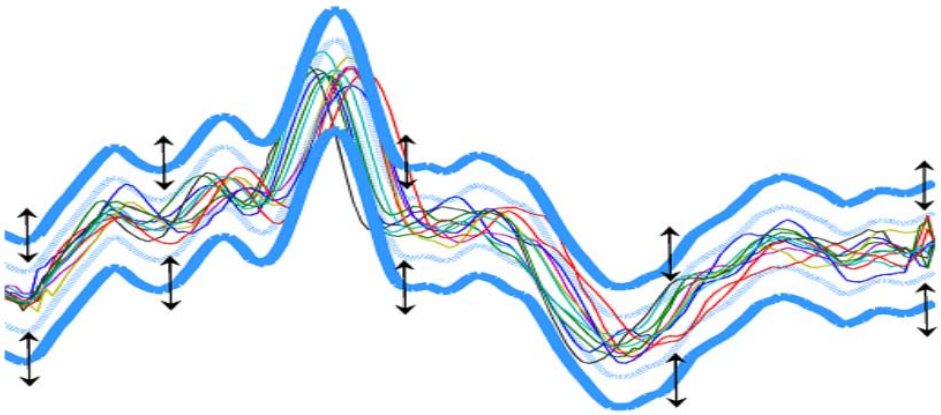


Figure 4.3: Intuition of an adjustable margin of similarity in motif discovery.

Approximate time series representations appear to be the best solution. Among these, the Symbolic Aggregate Approximation (SAX) has been shown to outperform other approaches (Ding et al., 2008). SAX has been widely used in the time series data mining community. The most important features of this approximation is that it reduces the dimensionality and lower bounds the true distance of the original

time series. Despite losing some of the information in the reduction process, it conserves the overall shape of the time series. The average calculation of the Piecewise Aggregate Approximation (PAA) is good against noise, except if sudden variations are the important aspect of an application domain. As shown in Lin et al. (2007) in the context of clustering, SAX sometimes outperforms Euclidean Distance on noisy data. This results from the smoothing caused by dimensionality reduction. SAX has been further enhanced to iSAX (Shieh and Keogh, 2008). The built-in multiresolution property makes iSAX even more attractive, since adjusting the margin of similarity to use i.e. increase or decrease the iSAX resolution, becomes a built-in feature.

As a symbolic approximation, SAX converts the original real time series T of length n into a sequence of w symbols – *word* – belonging to an alphabet of size $|\sum|$. The alphabet size of a given SAX word is called *resolution*. This operation is represented by the following notation: $SAX(T, w, \sum)$.

Figure 4.4 *a*) and *b*) depicts this idea for resolutions of 4 and 16, respectively.

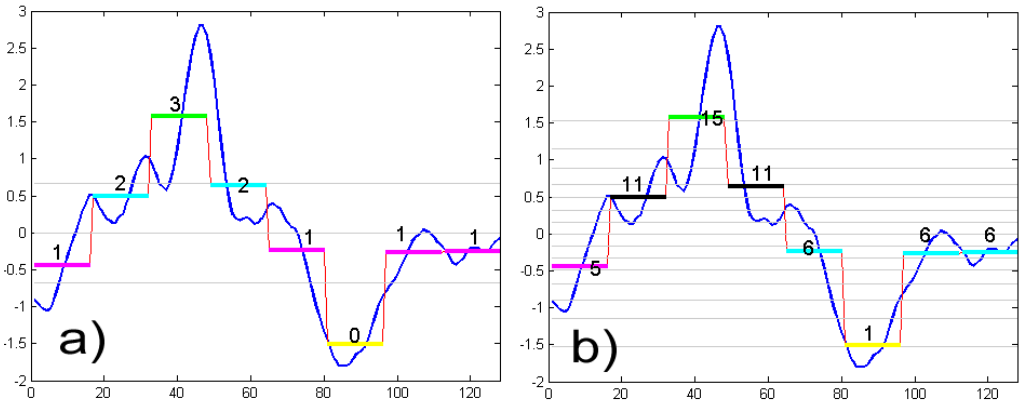


Figure 4.4: Example of the SAX conversion process for a time series with length 128, $w = 8$ and resolutions: *a*) 4, *b*) 16. Image generated by MATLAB and code provided by SAX authors Lin et al. (2003)

The iSAX representation extends classic SAX by allowing different resolutions for the same word. To avoid ambiguity, the resolution of each symbol needs to be made clear in the iSAX word. It is this enhancement that enables the creation of a time series index. For example, $w = \{2^4, 0^8, 7^8, 9^{16}\}$. However, for the scope of this thesis, we are not interested in the indexing capabilities of iSAX. Rather, we are interested

in interleaving between different resolutions within the same iSAX word.

Converting from a higher to a lower resolution is simple: one just needs to ignore one trailing bit as we reduce the resolution by half. However the opposite is not true, since one can have several possibilities for the higher resolution. We need to convert the original time series to the new resolution if we want the correct result.

An important issue to consider is that as the resolution increases, the more similar two time series need to be in order to originate the same word. Each interval narrows considerably each time we duplicate the resolution of the iSAX word. This intuition can be observed in figure 4.4, as we increase the resolution from 4 to 16.

For that reason, as we increase the resolution, finding a match for a given iSAX word becomes increasingly difficult. Our intuition is that at the largest resolutions (16 and 32), we will be working very close to the level of raw data (the size of the "blocks" get smaller and smaller). Time series need to be really similar to match the tiny 16 and 32 resolution blocks. For that reason, finding a match at a large resolution may indicate that the Euclidean Distance among the instances is very small. For example, with a word size of 8 at resolution 16, it is very unlikely that two time series generate the exact same iSAX symbol in all 8 symbols. Therefore we can skip expensive distance (e.g. Euclidean) calculations when we do have a match.

4.3 Background and Notation

In this section we introduce some notations and useful definitions.

Definition 4.1. A subsequence S with length m is an *instance of* an iSAX word W if $iSAX(S, w, a) = W$, for a given word length w and alphabet size a.

Both the w and a parameters do not need to be set by the user. Rather, they are part of the algorithm process as we shall see in section 4.4.2.

Definition 4.2. The *cluster* of the iSAX word W is the set of all instances of W in D.

We now clarify the goal of our data mining task. We aim to find the top-K motifs.

Definition 4.3. The $Top-K^{th}$ *Motif* of a time series database D is the cluster ranked at the k^{th} position regarding number of instances.

Notice that our motif definition does not consider a distance measure, as this is a very expensive step in motif discovery algorithms. The iSAX multiresolution capability allows to skip this step while maintaining the motif quality. In the next section further details are provided.

4.4 Our algorithm

In this section we describe the proposed algorithm. We start by briefly describing the space-saving algorithm for frequent items mining.

4.4.1 Space Saving algorithm

The Space-Saving (Metwally et al., 2005) (SS) algorithm was proposed to efficiently compute frequent elements in data streams. To the best of our knowledge, this algorithm has never been applied to time series motif discovery as it can not be directly applied to raw time series. The iSAX representation used by our algorithm outputs a discrete sequence of symbols, which is suitable to apply the SS algorithm.

Space-Saving is a relatively simple algorithm. Suppose we want to compute the top m elements of a data stream. The algorithm only needs to maintain m counters. These counters are updated such that the number of occurrences of the significant elements are accurately estimated. If the observed element e is in the monitored group then its frequency is incremented. Otherwise, the element e_m with the least estimated hits min in the monitored group is replaced by the observed element and the counter of that element is incremented. The algorithm's main goal is to never miss a frequent element. However, e can actually have between 1 and $min+1$ hits.

Algorithm 4.1 Space-Saving(m counters, stream S)

```
for each element, e, in S do
  if e is monitored then
    let count_i bet the counter of e
    Increment-Counter(count_i)
  else
    let e_m be the element with least hits, min
    Replace e_m with e
    Increment-Counter(count_m)
    Assign ε_m the value min
  end if
end for
```

Space-Saving is one of the most space efficient techniques for estimating top frequencies. Nevertheless, it is experimentally shown that monitoring only a moderate number of counters guarantees very small errors. Also, for each monitored element e_i, the maximum-overestimation ε_i for that element is saved, which is the value that the counter presented when the element was first inserted in the list. This provides an upper-bound in the over-estimation errors.

4.4.2 Multiresolution Motif Discovery

In this section we describe the Multiresolution Motif Discovery in Time Series algorithm – *MrMotif*. The algorithm is based on the iSAX representation. The main idea of the algorithm is to start by finding motifs at a low iSAX resolution (e.g. 4) and then continue motif discovery at higher resolutions (e.g. 16 or 32). As this expansion is performed, the number of instances of a given cluster reduces as each motif is split into several of the next resolution. Take for example, motif {0,1,1,2,4,6,6,7} at resolution 8, having 6 repetitions in the database. As we convert this motif to resolution 16, it is split into two distinct motifs: {1,2,3,4,9,12,12,15} and {1,3,3,5,9,13,13,14}. However, it is possible that no match could be found at resolution 16. If we do have a match at a high resolution, it means that the time series forming the motif are so similar that they could match at resolution 16 or 32.

At the highest resolution, a motif is formed only if the instances in that motif are very similar, as each iSAX symbol covers only a narrow interval in the amplitude of the time series. This idea can be observed in figure 4.4. This is the main intuition behind MrMotif.

For simplicity we assume the time series database D is available on disk. Regardless, each raw time series is not consulted more than once. Hence, the algorithm can be directly applied to streaming data. The minimum possible resolution g_{min} in iSAX is 2. The maximum resolution g_{max} is assigned to 64. Resolutions bigger than 64 are most of the time at the level of the raw series, where it is not possible to find motifs, or only spurious ones (identical time series). Note that we are not interested in analyzing all resolutions in between g_{min} and g_{max}. Rather, we only aim to study the g_{min} powers of 2 until we reach g_{max}. That is to say, we use the $2, 4, 8, 16, 32$ and 64 resolutions. In practice, we typically use only resolutions ranging from 4 to 32. These typically retain enough information. However, the user can select the desired resolution range. A set of hashtables *count* is maintained in main memory, one hashtable $count_g$ per resolution. Thus, pairs of $(motif, count)$ are stored.

Our algorithm aims to find the solution for the following problem:

Problem definition: Given a time series database D, a motif length m and a K parameter, for each resolution in $(g_{min}, g_{min} \times 2, \ldots, g_{max})$ find the top-K motifs.

We describe the pseudo-code in Algorithm 4.2. The actual implementation can be accessed at Castro (2010). For simplicity, we describe the algorithm without considering details about memory usage or cluster hierarchy. We will detail later on when exactly the Space-Saving setting is activated and how the information that will allow visual tools to navigate through the motif structure is saved. For the time being, assume Space Saving is not active (line 9), and ignore line 3. The algorithm is relatively straightforward. A sliding window of size m is used to scan the subsequences of all time series T_i (with possibly different sizes) in database D. Also a bounded buffer of size m is used to keep this disk traversal sequential. We are aware that contiguous subsequences are likely to be almost identical and for

Algorithm 4.2 MrMotif(D, m, K)

```
1:  for each subsequence S of length m in D do
2:      W ← iSAX(S, g_min ... g_max, w)
3:      motifTree.Update(W)
4:      for each w_g in W do
5:          if w_g is in count_g then
6:              c_g ← count_g.get(w_g)
7:              count_g.Update(w_g, c_g + 1)
8:          else
9:              if Space-Saving is Active then
10:                 (e_m, min) ← count_g.getMin()
11:                 count_g.Update(w_g, min + 1)
12:                 ε_m = min
13:                 count_g.updateMinimum()
14:             else
15:                 count_g.Update(w_g, 1)
16:             end if
17:         end if
18:     end for
19: end for
20: return count
```

that reason a step greater than one is used in the sliding window approach. This prevents spurious motifs from being found, also known in the literature as "trivial matches" (Lin et al., 2002). Each m-length subsequence read is converted to an iSAX word for each resolution of interest (line 2). Note that the conversion of a subsequence to all resolutions is executed in one single step, as for the same time series most of the conversion process is similar at all resolutions (only the symbol lookup is independent). Then, if the motif exists in the corresponding hashtable $count_g$ structure, its count is incremented and the location of the subsequence saved (lines $4-7$). Otherwise it is set to one (line 15). Finally, the top-K Motifs for each resolution are outputted (line 20).

Space-Saving

In section 4.4.1 we have described the Space-Saving algorithm without referring when it is activated and which elements to monitor. The intuition is to directly apply it to the Top-K motif problem. However, this would not yield satisfactory results because this K set is typically very small (for instance, Top-10). Thus, it could potentiate the number of over-estimation errors. Instead, we let the user decide the maximum amount of memory the algorithm's implementation has available. The amount of memory the algorithm is using is monitored. If the algorithm reaches the user defined threshold the Space-Saving mode is activated. In that case, the algorithm will execute lines 9–13. For example, in a mobile device this threshold can be set to 1 MB. It is also possible that the user chooses not to set this threshold. In that case the algorithm is executed until no more memory is available. As this situation can make a system stop operation, we actually "hard-code" this threshold to 99% of the available memory. One could argue that by using a clever representation technique as we do, this will hardly occur. However, that is not the case. We will show this situation using a relatively small time series where the number of different motifs in the counters hashtable increases at a fast rate. This occurs specially at larger resolutions, since less matches occur, and we must store all words in the hashtable. On the other hand, having initialized the system in "full memory" mode provides us a large enough number of counters to ensure very small errors. The use of ε_m provides guarantees about the quality of a given execution of the algorithm.

Interactive Visual Tool

In section 4.1 we have discussed that data mining algorithms should provide rich outputs. This would facilitate the development of applications that receive as input that same output, such as Visualization applications (Lin et al., 2004). Hereby we

show an example of such application, as we believe these provide the data mining practitioner with further understanding and intuition about the data at hand.

Our example application is the iMotifs Interactive Time Series Motif Discovery and Visualization Tool (Castro, 2011a), which allows to perform several exploration and "drill-down" operations along the motif hierarchy. During the algorithm's execution, an iSAX word is generated for each subsequence within each resolution. Larger resolution words are contained in smaller ones. In this sense we say that the discovered motifs form an hierarchy. For example, the length 2 word $(4^8, 3^8)$ contains the words $(9^{16}, 7^{16})$ and $(8^{16}, 7^{16})$. We say that the lower resolution is the parent motif and the higher is the child. The words generated for the same subsequence form a word family. It is straightforward to keep and maintain this information in a tree structure. Line 3 of the algorithm performs this maintenance operation. The motif tree structure can then be input to a graphical user interface in the form of a graphical tree (similar to a file system tree). The user can explore and visualize the motifs at different resolutions, in order to realize which of the frequent motifs are significant for his particular domain/problem. Figure 4.5 displays a screenshot of the iMotifs navigator where the motif hierarchy structure can be observed.

4.5 Experimental Analysis

In this section we perform experiments to validate the impact of the proposed algorithm. First the space and time scalability is analyzed in a large synthetic database. Then, the effect of noise in the algorithm is studied. Finally, the impact of MrMotif is shown in three different real applications. The experiments were performed on a machine with a Quad-Core AMD Opteron^{TM} Processor 2352 with 16 GB of RAM. The MrMotif algorithm is implemented in Java and the compiler used was the JDK 6.

Figure 4.5: Snapshot of a motif navigator

4.5.1 Scalability Experiments

The experimental analysis begins by considering the scalability of the proposed algorithm. We compare our approach to Random Projection (RP) (Chiu et al., 2003) in terms of execution time. This algorithm is selected for comparison due to its popularity. It is the most cited time series motif discovery proposal up to date (more than 240 citations) and is the basis of many current approaches that tackle this problem (Minnen et al., 2007a; Tanaka et al., 2005; Yankov et al., 2007a), as studied in our survey 3. Furthermore, the execution time of this algorithm can be used as a lower bound on the execution time of all approaches that are based on it. We also perform comparisons with the "full memory" (FM) version of our algorithm in order to understand the impact of Space-Saving (SS). The dataset used in this section is constituted by random walk time series available in the MK algorithm (Mueen et al., 2009b) website. We select these data for two reasons: they have been used before and results on similar datasets are encouraged in order to walk towards

data mining benchmarks; also, the size (in the Gigabytes order of magnitude) makes them attractive to test any algorithm. The dataset is composed by ten different sets of random walk series, with 10000 to 100000 time series of length 1024. These sets occupy a large amount of disk space ranging from 160 MB to 1.5 GB, for a total of about 8 GB in the database. We reproduce these datasets by following the instructions in (Mueen et al., 2009b) website, using the same random seed as MK's authors. In the MK proposal, the algorithm is executed 10 times for each of the ten increasingly large datasets and the average of the execution time for each dataset is recorded. We follow the same approach with RP and both implementations of MrMotif – SS and FM.

The motif discovery algorithms are executed with $K = 10$ and $m = 1024$. We follow the recommendation of the SAX authors (Lin et al., 2003) and set $w = 8$ (iSAX word size) for all experiments. The maximum amount of memory used by SS is set to 128 MB. This value is selected because it is close to the average RAM available in current mobile phones. We implement the RP algorithm and set the parameters $w = 8$ and $a = 8$. The c parameter is randomly chosen between 2 and 7, to assure that the distribution of the projections is wide enough to prevent the algorithm from becoming quadratic. For fairness, we remove the disk verification of candidate motifs (a module part of RP), since MrMotif does not perform this expensive step. Both implementations of MrMotif and RP are available on MrMotif website (Castro, 2010). Figure 4.6 displays the results of the execution.

It can be observed that MrMotif is about one order of magnitude faster than RP for each of the ten increasingly sized sets that constitute our dataset. Also, MrMotif execution time increases linearly as the dataset size increases, as expected. However, RP seems to grow quadratically. The reason for this is that RP is quadratic with respect to the SAX word list size. Note that we present results for just one iteration of RP. However, as an iterative algorithm, several iterations are necessary in order to converge. The reason we show results for one iteration is to make clear that MrMotif full execution outperforms one iteration of RP. A full execution of MrMotif returns the top-10 patterns for the 2, 4, 8, 16, 32, and 64 resolutions deterministically. It

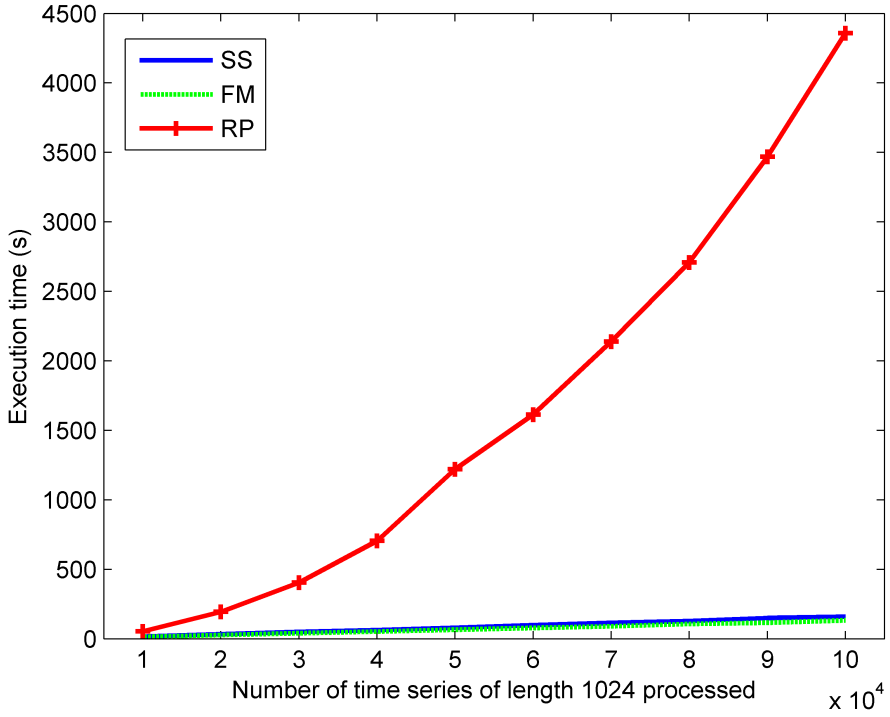


Figure 4.6: Variation of the execution times of the three algorithms as the number of processed time series increase.

can also be observed that the FM version of MrMotif executes faster than the SS version. This is due to a small overhead that Space-Saving adds to the algorithm.

The next experiment reports the memory usage of the MrMotif SS and FM versions during the execution in the dataset containing 100000 time series of length 1024. The MrMotif SS and FM versions are compared in this experiment in order to show the impact of Space-Saving. Figure 4.7 depicts the memory used by the Java Virtual Machine of the FM and SS algorithms versions.

It can be observed (as expected) that when the SS algorithm is activated (at time series 6000), the memory used by the algorithm remains below the imposed limit. The wave-form of the memory usage variation can be explained by the effect of the Java garbage collection (GC). The FM version of the algorithm uses a large amount of memory. This is due to the fact that 100000 random time series produce a large number of iSAX words, which quickly fill the hashtables. This also happens with RP or any other algorithm that saves the iSAX representation of all time series in

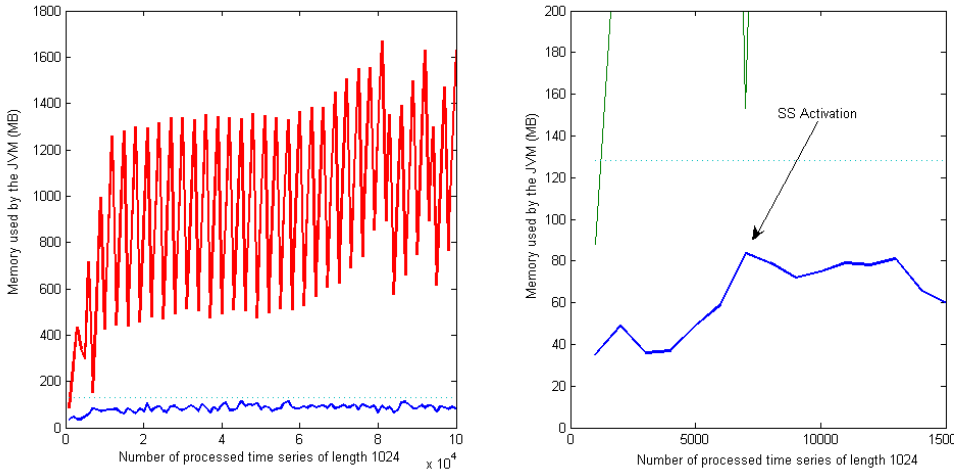


Figure 4.7: Variation of the memory used by the JVM as the number of processed time series increase. *Red*: FM, *Blue*: SS. The *right* figure zooms in the left bottom quadrant of the chart.

main memory. However, the FM version is only used for experimentation purposes. In real scenarios, we use the SS approach.

In this section we have demonstrated that MrMotif SS version is time and space efficient for relatively large datasets (8 GB). The results show that the algorithm is linear regarding the number of time series in the database. This is due to the use of a single sequential disk traversal and constant access time structures (hashtables). It is also observed that the proposed algorithm executes about one order of magnitude faster than RP. This is an encouraging result because RP is the basis of many existing motif discovery algorithms.

4.5.2 Experiments with noise

In this section an analysis of the impact of noise in MrMotif results is performed. We start by applying MrMotif to the set of 10000 time series of length 1024 from the previous experiment. We record the top-10 patterns of resolution 4 and use these results as the ground truth for our study. Then, we produce ten noisy variations of our dataset using the technique (and code) in Vlachos et al. (2003): Gaussian noise and small time warping is added to the original series. Further details of the

technique can be accessed in the original paper (Vlachos et al., 2003) or in our website (Castro, 2010). For each variation we increase the range of noise introduced, from 10% up to 100% of the original series standard deviation. We apply our algorithm to each of the ten noisy versions of our dataset, recording the information retrieval metrics *precision* and *recall* of each execution with respect to the original (noise free) version. These measures are calculated by using the number of true positives (TP), false positives (FP), and false negatives (FN) for each execution: the TP are the number of *clusters* present in the top-10 of the noisy and original version; the FP are the number of clusters that are incorrectly in the noisy version; and the FN are the clusters that are not in the noisy dataset but are in the original execution's. We have Precision $= TP/(TP + FP)$ and Recall $= TP/(TP + FN)$. The results are shown in figure 4.8.

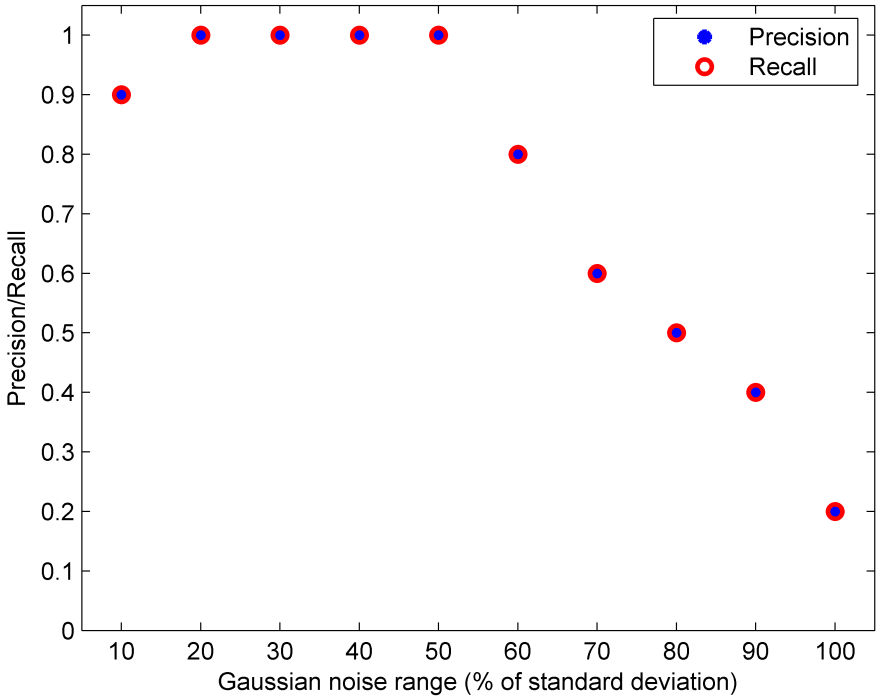


Figure 4.8: Variation of the Precision and Recall of each increasingly noisier variation of the original 10000 size dataset.

We can observe that precision and recall present values above 90% until the noise range is greater than half standard deviation. From this point onwards, the noise level causes the series to significantly differ from the original ones. We can conclude

that MrMotif is robust to relatively high levels of noise and small time warping. For these levels, few of the top-10 patterns were missed and a small number of false patterns in the top were discovered. This capability is derived from the smoothing effect of the iSAX dimensionality reduction process. In our experiment setting a fixed value for the top patterns (10) is used. For that reason, when our algorithm misses a top pattern (FN) it obviously introduces a spurious one in the top-10 (FP) and vice-versa. Therefore, precision and recall present the same value for this experiment scenario.

4.5.3 Real Applications

In this section the MrMotif algorithm is applied to three different application areas. Our goal is to validate that MrMotif is a strong candidate for a wide range of applications where time and space efficiency are necessary. Our algorithm is applied to real datasets from the areas of protein unfolding, sensor networks monitoring and telecommunication networks.

Protein Unfolding

Protein folding involves the formation of the 3D structure of a protein from a sequence of aminoacids. Folding or unfolding disorders of a protein causes diseases such as the neurodegenerative Alzheimer's. The Transthyretin (TTR) is one example of such proteins whose unfolding disorders cause severe diseases. Unfolding mechanisms of this protein have been studied by computationally analyzing variations on certain molecular properties over time. The Solvent Accessible Surface Area (SASA) is one example of such properties that are important to study in order to understand the cause of the disorder and consequent manifestation. This analysis is performed by means of simulation from Molecular Dynamics (MD) unfolding of TTR

(Azevedo et al., 2005). The dataset is constituted by 127 time series of 10000 points each, corresponding to the variation of the SASA in each of the 127 aminoacids of the protein during a period of 10 nanoseconds (ns): one point per picosecond (ps) of simulation. The actual time necessary to run this $10ns$ simulation surpasses one month. We apply our proposed algorithm to find the Top-10 patterns of size 64 ($64ps$), as this may unveil important repetitions in unfolding behavior among the different aminoacids. The top-1 motif retrieved by our algorithm is discussed. The larger resolution cluster with at least 2 repetitions was discovered at the 16 resolution. The top clusters at resolution 8, 4, and 2 presented 5, 35, and 1029 instances, respectively. In figure 4.9 one example of a motif found at the resolution 4 is displayed.

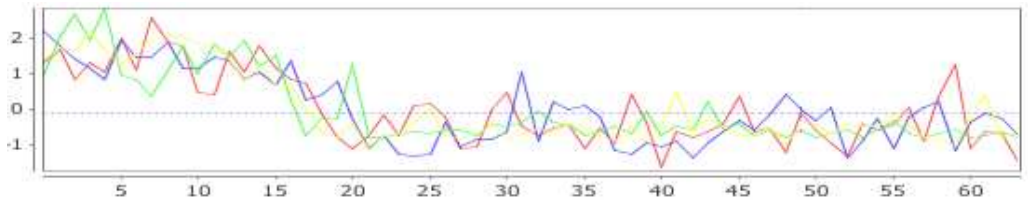


Figure 4.9: The four instances of a motif discovered at resolution 4.

The discovered motif is repeated 4 times in the database. This motif highlights the robustness to noise characteristic of our algorithm. The time series instances are not exactly equal and present a relatively large Euclidean Distance. Nevertheless, they were successfully counted as repetitions. This capability of MrMotif provides the biologists with further insight on the domain.

Sensor Networks Monitoring

To apply data mining techniques to emerging architectures such as sensor networks is of particular importance. These devices will be widely used in the future in fields as diverse as health, forest fire detection, and general surveillance. Sensors communicate with the sensor base through wireless channels. These operate at a frequency close to the Wifi networks and for that reason are subject to interference

and failures. It is then vital to monitor parameters of the sensor networks communication protocols, such as the delay or number of retransmitted packets caused by packet loss. The reason is that a packet loss will cause a retransmission, which is the most energy-expensive operation for these battery-run devices. Our dataset is composed by averaged delay data of a specific sensor in a wireless network of biomedical sensors. There are 9 time series, each covers a monitor period ranging from 7 hours up to 18 consecutive hours. Each data point contains the average delay of packet transmission by the iLPRT MAC protocol (Gama et al., 2009). MrMotif is applied to the dataset to find the Top-5 motifs, using a motif length of 16 covering the last 16 minutes, as suggested by the domain expert. The memory limit is set to 1 MB for this particular scenario. This highlights the amount of memory these devices typically have available. Figure 4.10 shows one example of a motif detected at resolution 8.

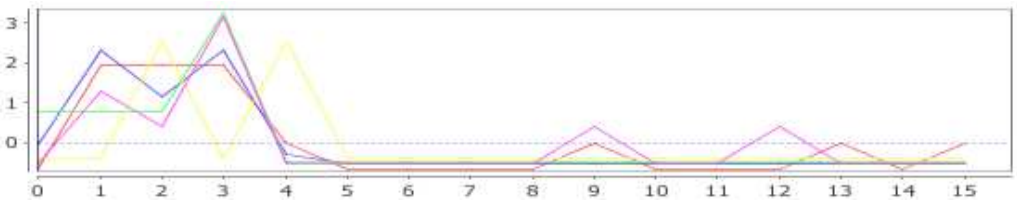


Figure 4.10: Example of a motif with 5 instances. The variation was due to interference by a laptop's antenna wireless.

This motif presents 5 time series repetitions. The motif has been acknowledged by the domain expert as worthy of further investigation. The displayed motif occurred in situations where the user intentionally approached a laptop to the sensors range in order to cause interference. This provides promising results in further applications of MrMotif and other time series data mining algorithms. The goal is to help improve communication protocols for wireless sensor networks.

Telecommunication Networks

Telecommunications networks interconnect people of different cities, countries and continents. For that reason they play a central role in nowadays society. These

networks are characterized by very complex and large structures that are monitored by network operators, using reports of performance counters such as traffic data. For network troubleshooting problems it is interesting to detect frequent patterns. This helps in preventing future failures, obtaining further knowledge about the domain, and achieve better next generation networks. In this experiment MrMotif is applied to a traffic dataset from a Portuguese telecommunications network operator. The data regards several network elements (nodes), whose traffic was recorded in the period of a week at a granularity of 15 minutes. The algorithm parameters K and m were set to 10 and 360, respectively. The goal of the network operator is to attempt to find regularities in the network traffic, possibly at different nodes, over the period of a few days. In figure 4.11 one example of a motif with two instances returned by our algorithm at resolution 8 can be observed.

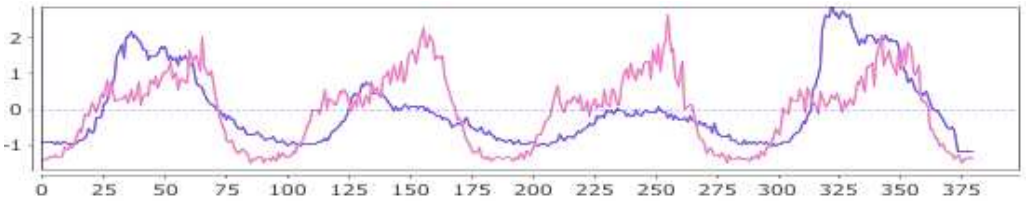


Figure 4.11: Motif with two instances found at two different nodes.

The network operator has found this result and approach interesting. These motif represents the same characteristics in telecommunications traffic at a given week and the causes remain to be investigated by the telecom operator.

4.6 Conclusion and Future Work

We have proposed the Multiresolution Motif Discovery in Time Series algorithm – MrMotif. This proposal tackles limitations of existing algorithms such as disk access and memory inefficiency. It brings to time series motif discovery the Space-Saving algorithm in order to efficiently handle strict memory requirements present in emerging architectures as sensor networks. The multiresolution property inherited by the solid iSAX representation allows to find motifs at different resolutions, which

avoids the expensive motif verification step in the raw data. This provides useful insight to the practitioner about the database at hand. MrMotif is scalable and can have a strong impact on different application areas due to the good performance and robustness to noise. Future work includes investigating time series motifs evaluation measures (Ferreira and Azevedo, 2007) and studying automatic methods to derive the parameters K and m.

- 5 -

Time Series Motifs Statistical Significance

5.1 Introduction

Since the problem formulation in (Lin et al., 2002), many proposals on how to extract motifs from a time series database have been introduced (Castro and Azevedo, 2010; Chiu et al., 2003; Ferreira et al., 2006; Minnen et al., 2007a; Mörchen and Ultsch, 2007; Mueen and Keogh, 2010; Mueen et al., 2009a,b; Oates, 2002; Tanaka et al., 2005; Yankov et al., 2007a). Surprisingly, most of these proposals do not focus on how to evaluate the extracted motifs. Returned motifs tend to be subjectively evaluated by humans because they are application dependent and not previously labeled – motif discovery is an unsupervised task. In practice, this is unfeasible. Datasets are often large and motif mining algorithms typically return a prohibitively large number of patterns. To restrain to expert analysis the most frequent motifs is not an interesting approach, as frequent patterns are not necessarily the most interesting ones. Many frequent patterns are spurious, trivial or simply expected: they are not meaningful to the user. In a randomly generated database of length 65536 from (Keogh and Folias, 2002), for example, 65 motifs are discovered. The top motif reaches 4 repetitions, and the average motif count is 2.17. Since a random process generated the database, all discovered motifs are meaningless. This example highlights the need for automatic time series motifs evaluation.

Statistical tests have been successfully applied to other pattern mining problems.

For example, in bioinformatics they have been used to detect DNA segments with significantly unexpected frequency (Robin et al., 2007); in networks analysis, to find significant subgraphs (Milo et al., 2002); in association rules mining to discard redundant rules (Webb, 2007). In all these examples the common question to be addressed is: "Can this pattern be observed so many times just by chance?". These approaches consider the observed count (frequency) of a pattern which is typically compared to its expected count. This difference is then statistically analyzed. However, this method cannot be directly applied to time series data since it is not clear how to calculate the expected frequency of a given section of the series.

To overcome this limitation and take advantage of the wealth of available algorithms for symbolic data (DNA sequences, text, etc.), we use a symbolic definition of time series motifs. Our approach is based on work from bioinformatics (Robin et al., 2007). We estimate the probability of occurrence of a word (motif) using Markov Chain Models. In these models, the probability of a motif is estimated according to its subword count. Given a motif, we compare the difference between its observed count and estimated expected count in terms of statistical significance. Namely, we calculate the *p-value* of this difference, aiming to answer whether we can observe such a count solely by chance.

Our contributions are twofold: to provide an approach to assess the statistical significance of time series motifs, and to compare the performance of several simple statistical hypothesis tests on motifs extracted from real datasets. The novelty of our work is that it enables the calculation of time series motifs p-values. To the best of our knowledge, this has not been attempted in the literature. It has been shown to be an important problem in DNA, protein, and network motifs (discrete motifs). We provide the link between the well studied discrete motif significance problem and time series motif evaluation. This allows time series data mining practitioners to evaluate better the motifs extracted from their data. It also provides researchers with a method to evaluate properly the output of motif discovery algorithms using statistical significance.

The remainder of the chapter is organized as follows: section 5.2 describes the state

of the art in motif statistical significance; background and notation used throughout the chapter are described in section 5.3; in section 5.4 an approach for assessing time series motifs significance is proposed; the experimental analysis is described in section 5.5; finally, in section 5.7 we derive conclusions.

5.2 Related Work

Since the introduction of the time series motif discovery problem in Lin et al. (2002), many approaches have been proposed (Castro and Azevedo, 2010; Ferreira et al., 2006; Minnen et al., 2007a; Mörchen and Ultsch, 2007; Mueen and Keogh, 2010; Mueen et al., 2009a,b; Oates, 2002; Tanaka et al., 2005; Yankov et al., 2007a). Most of these works tackle the algorithmic details of the motif extraction process. Surprisingly, the critical aspect of evaluating the extracted motifs has not received much attention by researchers. The results are typically interpreted by experts on the domain at hand. This approach is untenable for large real-world datasets that can reach terabytes of data. Automatic motif evaluation procedures are required.

According to Ferreira and Azevedo (2007), motif mining evaluation measures can be classified in the following categories: class-based, theoretic-information, mixed measures and statistical significance tests. Class-based measures (accuracy related) are calculated by comparing the motif occurrences with the ground truth using a confusion matrix. Examples are precision, recall and specificity. Theoretic-information measures are calculated using probabilistic or information criteria contained in the motif itself. Examples are the Information Gain and the Minimum Description Length. Measures such as Mutual Information and J-measure are mixed, because they use both class-based and theoretic information criteria. From this set of measures, we are particularly interested in statistical significance tests. These tests are very popular in science in general and data mining in particular. They tend to be accepted as the *de facto* standard to evaluate significance or help in the decision making process.

Statistical significance tests are widely used in bioinformatics. Without claiming to be exhaustive we mention a few of these works. Zhang et al. (2007) define the problem of evaluating statistical significance of DNA motifs as the ranking of such motifs according to an underlying model, defined using Markov chains. A dynamic programming algorithm (MotifRank) is proposed to compute motif exact p-values. Marschall and Rahmann (2009) propose a methodology to calculate p-values with respect to independent and identically-distributed (i.i.d.) and Markov models. A compound Poisson approximation is used for the number of motif occurrences (null distribution). These techniques are integrated in an efficient motif discovery algorithm by exploiting the monotonicity property of the compound Poisson approximation. The algorithm is applied to IUPAC strings (chemical compounds representation) and *Mycobacterium tuberculosis* data. Nuel (2006) provides recursive algorithms to compute Cumulative Distribution Functions (CDF) using Finite Markov Chain Imbedding (FMCI). The algorithms are applied to discover exact p-values of patterns aiming to find hydrophobic segments in protein data. In Boeva et al. (2007), the authors introduce an algorithm to calculate the probability of finding multiple occurrences of motif in a random text. This probability is calculated using both the Bernoulli and order one Markov chain models. The approach is applied to find the statistical significance of binding sites frequency in regulatory modules of eukaryotic genes. Low Kam et al. (2000) propose an algorithm to mine unexpected frequent sequential patterns in DNA and protein sequences. Sequential patterns are defined according to a Markov model and patterns support follow a Binomial distribution. The p-values that measure over-representation are then calculated. Hollunder et al. (2007) introduce the DASS algorithm to estimate the statistical significance of patterns in protein data. Several techniques for determining the expected value of each pattern such as data permutations, shuffling, and the binomial distribution are used. Robin and Schbath (2001) perform an experimental comparison of several distributions of word counts in random sequences, regarding accuracy and computational cost. The exact distribution is compared to the Gaussian and compound Poisson approximations in the extraction of exceptional words of the phage *Lambda* genome. In Régnier and Vandenbogaert (2006), the drawbacks

of the Gaussian approximation are analyzed. Schbath (Schbath, 2000) studies the statistical distributions of word counts in Markov chains. Formulae are derived for the estimated expected counts under these distributions. In Robin et al. (2007), statistical tests are used to compare motif count exceptionalities in two (or more) sequences. The exact binomial and the asymptotic likelihood ratio test are used. The motif count is modelled using Poisson processes. The motifs in the backbone and loops of the *Escherichia coli* K-12 bacterium are compared.

In the networks (graph) mining community, the issue of statistical significance in motif discovery has also received much attention. In He and Singh (2006), a Binomial test is used to evaluate the statistical significance of frequent subgraphs in a chemical compounds graphs database. Milo et al. (2002) define network motifs as patterns of interconnections with a significantly higher frequency than those in randomized networks, according to their Z-score. A comprehensive experimental analysis is done in complex networks from biochemistry, neurobiology, ecology and engineering. In Jacquemont et al. (2009) the authors convert sequential data to probabilistic automata and then integrate statistical constraints to reduce the search space of the exploratory process. The approach is applied to car flow modelling data. Ribeca and Raineri (2008) derive a fast motif Z-scores exact calculation method using discrete finite-state automata (DFA), assuming the sequence is generated by a Markov model of arbitrary order. The authors experimentally test their approach in large scale human genome and yeast binding factors data. Matias et al. (2006) provide exact formulas for the expectation and variance of a motif's number of occurrences. This approach also introduces a simple and efficient probabilistic model for the motif distribution in networks, which is much more efficient than the traditional comparison to randomized (simulated) networks. In Picard et al. (2008), the authors consolidate a decade of research in biosequences motifs exceptionality and apply it to the network motifs scenario. Several motif distributions approximations are compared such as the compound Poisson distribution and the Gaussian approximation. Approximate p-values are calculated to assess the exceptionality of observed motif counts. The method is applied to protein-protein interaction networks.

There is a handful number of time series motif mining proposals that consider the significance evaluation aspect of extracting motifs. Ferreira et al. (2006) use the Information Gain and Log-Odds measures to assess the statistical significance of motifs. However, the order dependency (time) that characterizes time series data is not taken into account. In Chiu et al. (2003), the authors use a statistical test as a criterion to stop their iterative motif discovery algorithm, i.e. the algorithm ends the execution when the observed motif count significantly exceeds the expected by chance. In this work, we aim to go one step further and calculate each motif's p-value according to their statistical significance. In the context of time series anomaly detection, Keogh et al. (2002) propose an approach to find surprising patterns in time series data. Markov Chain Models are used to predict the expected frequency of patterns, given a collection of previously observed normal data. However, the motif discovery problem is unsupervised. It is not possible to know beforehand which patterns are significant. Moreover, we are not interested in finding anomalous patterns. Rather, we aim at statistically stating which frequent patterns are also significant by calculating each pattern's p-value.

5.3 Background and Notation

In this section we introduce some notations and useful definitions.

For the scope of this work, all time series are normalized in order to remove offset and scaling effects. It has been shown that comparing time series that are not normalized is meaningless (Keogh and Kasetty, 2003). For simplicity, we treat each subsequence of database as a different time series in D. In practice, to slide a window through a long time series for the purpose of extracting subsequences is similar to handling each subsequence as a different time series. Possible motif overlaps are handled by taking into account trivial matches (Chiu et al., 2003). Since our work is inspired by the biosequences motif mining, we are interested in the symbolic representation of time series and their subsequences.

Definition 5.1. A *word* $w = w_1 w_2 \dots w_l$ is the symbolic representation of a subsequence S, with $w_i \in \Sigma$. The Σ is the representation alphabet and its size is named the representation *resolution*.

The symbolization of S by a generic times series representation technique R is denoted by $R(S) = w$. In this work we use the best representation technique available in the literature for time series data, as experimentally shown in Ding et al. (2008). The Symbolic Aggregate Approximation (iSAX) (Shieh and Keogh, 2008), representation takes as input a time series and transforms it into a sequence of symbols, as highlighted in Fig. 5.1.

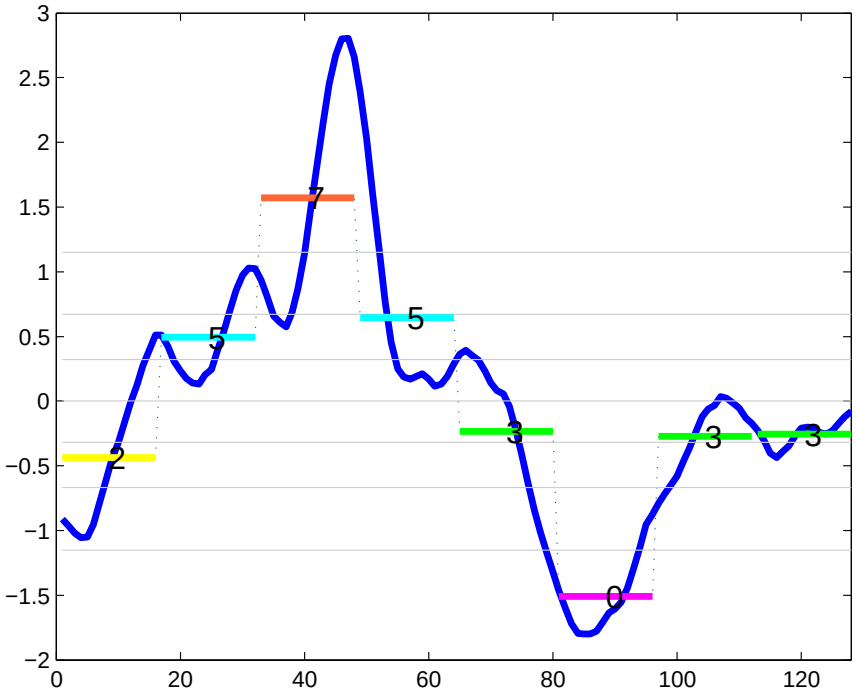


Figure 5.1: Conversion of a time series into its iSAX representation, generating word $\{2, 5, 7, 5, 3, 0, 3, 3\}$.

As depicted in figure 5.2, slightly different subsequences can originate the same word in the representation space. These subsequences are called instances of the given word.

Definition 5.2. A subsequence S is an *instance* of a word w if $R(S) = w$, where $R(S)$ is a symbolic representation of S.

Matching between two or more instances of word w is defined as follows:

Definition 5.3. Subsequences S_1 and S_2 *match* if their symbolic representations are the same, i.e. $R(S_1) = R(S_2)$.

At this point, we are ready to formalize the notion of time series motif. An example of a motif with 3 instances is shown in figure 5.2.

Definition 5.4. The word w is a *Motif* in database D if the count of all instances of w in D is greater than 1.

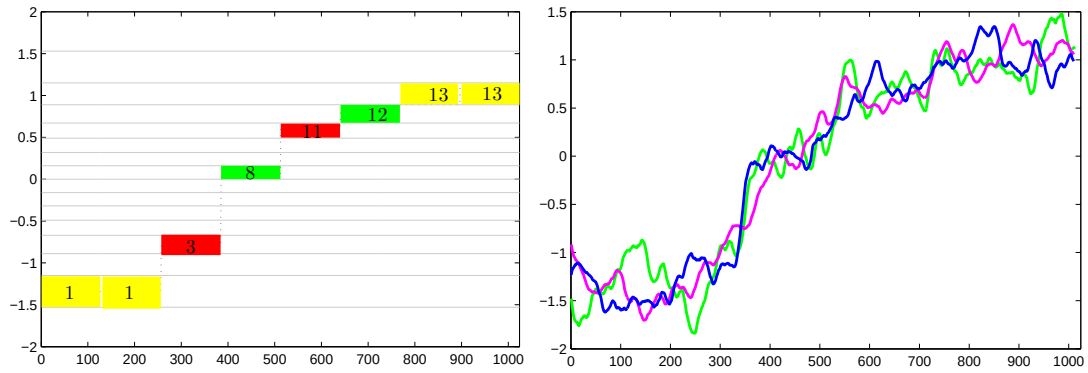


Figure 5.2: Motif $\{1, 1, 3, 8, 11, 12, 13, 13\}$ (left) and its 3 instances in the database (right).

Definition 5.5. The *motif count* (or frequency) of a motif M is the total number of instances M has in database D.

Time series motifs are typically sorted according to their motif count.

5.4 Time Series Motifs Statistical Significance

In this section we introduce an approach to assess the statistical significance of time series motifs. To the best of our knowledge, there is no approach available in the literature to calculate the p-value of time series motifs. Our approach methodology is described next. First, motifs are extracted from the database. Second, the

probability of each motif is calculated using Markov Chain models. Statistical hypothesis tests are then applied according to several distributions for the motif counts (Binomial, Poisson and Gaussian distributions) to calculate each motif's p-value. In this section the false discovery rate problem is also considered.

5.4.1 Extracting Motifs

The first step of the motif significance evaluation is the actual extraction of frequent motifs. There is a plethora of time series motif discovery algorithms in the literature (see section 5.2). Among those, exact algorithms (Mueen et al., 2009b) have been shown to be a sound contribution to the time series motif discovery problem. Despite being less accurate than their exact counterparts, approximate algorithms present a relatively good trade-off between accuracy and efficiency. They are also typically robust to noise (Castro and Azevedo, 2010; Chiu et al., 2003). In this work, to leverage the existing work in bioinformatics motif discovery, we are interested in symbolic motifs, i.e. discretized representations of the discovered motifs. Therefore, we select an approximate algorithm, that internally uses a symbolic representation and outputs discrete motifs. It is noteworthy that any motif discovery algorithm can be used, since its output is symbolic, or discretized using iSAX. The recently introduced MrMotif (Castro and Azevedo, 2010) is an excellent candidate to represent the symbolic motifs approach. It uses a symbolic definition of time series motifs, a necessary property to take advantage of the wealth of existing work in the bioinformatics. It also outputs the most frequent motifs in a straightforward manner (a list of words) and it is efficient (linear complexity). MrMotif takes as input a time series database D and a parameter K and derives the top-K motifs in D and their count. This step is shown in figure 5.3.

For simplicity, we choose to evaluate motif statistical significance as post processing task. This process can also be integrated in the motif search itself as demonstrated in Jacquemont et al. (2009); Marschall and Rahmann (2009); Zhang et al. (2007).

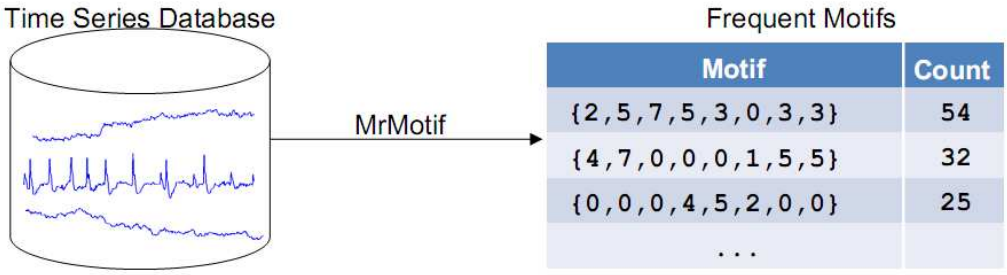


Figure 5.3: Extraction of frequent motifs from the time series database.

5.4.2 Reference Model

By its own, motif support is not a good interestingness measure. Similarly to support in itemsets mining, frequency does not guarantee that motifs are significant. A trivial example highlighting this problem is shown in section 5.1 (random time series data are used). This example shows that patterns can be found to be frequent even in random data. However, those frequent patterns are meaningless because they were randomly generated. A better approach is to consider the difference between the observed motif count and the motif expected count. The expected count is the number of motifs one should expect in random sequences similar to our database (under some similarity definition). This knowledge is obtained considering a reference model that reflects the background distribution of the motifs.

Random sequences are typically modeled using Bernoulli trials or Markovian sequences (Schbath, 2000). The former assume that words are i.i.d., although word symbols are possibly neither independent nor identically distributed in real data (Schbath, 2006). The latter take the composition of the words into account. That is, they consider the time dependency characteristic of time series data. Also, there are analytical probability calculations available which prevents expensive simulations (Schbath, 2006). A Markov chain is a mathematical system to model random processes. It is composed by states and the transitions between them in a chainlike fashion. That makes them suitable to model sequential symbolic data, where each state is a symbol in the sequence. They have been widely used (He and Singh, 2006; Matias et al., 2006; Milo et al., 2002; Nuel, 2006; Ribeca and Raineri, 2008; Schbath,

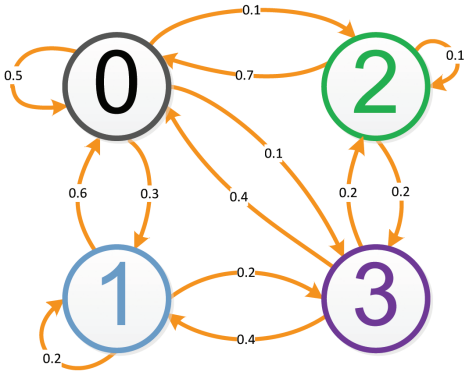


Figure 5.4: Example of a Markov chain.

2000). A simple Markov chain is shown in figure 5.4.

The time dependency is considered by assuming that the distribution of each symbol depends on its previous symbols. In particular, each symbol depends on the symbols that immediately precede it, rather than its entire past (Markov property) (Robin et al., 2005). Given a random sequence of symbols from an alphabet $\mathscr{A} = \{0, 1, 2, 3\}$:

$$S = (X_1, X_2, \ldots, X_i, \ldots)$$

the probability of each symbol depends only on the recent past of the sequence, i.e. it depends only on the symbols $(X_{i-m}, \ldots, X_{i-1})$. The integer m is the order of the Markov chain model. A Markov chain model of order m is denoted by Mm. For example, in model M1, each symbol depends only on the previous one. This dependence is expressed by the probability to transit from one symbol to the next. This probability is the transition probability $\pi(x, y)$, where $x, y \in \mathscr{A}$. Formally, the probability that X_i is at state y, given X_{i-1} is at state x is:

$$\pi(x, y) = \mathbb{P}\{X_i = y | X_{i-1} = x\}$$

The set of all transition probabilities form a transition matrix (Robin et al., 2005). For example, the transition matrix for $\mathscr{A}$ is:

$$\Pi = \begin{pmatrix} \pi(0,0) & \pi(0,1) & \pi(0,2) & \pi(0,3) \\ \pi(1,0) & \pi(1,1) & \pi(1,2) & \pi(1,3) \\ \pi(2,0) & \pi(2,1) & \pi(2,2) & \pi(2,3) \\ \pi(3,0) & \pi(3,1) & \pi(3,2) & \pi(3,3) \end{pmatrix}$$

Considering the Markov chain in figure 5.4, we have:

$$\Pi = \begin{pmatrix} 0.5 & 0.3 & 0.1 & 0.1 \\ 0.6 & 0.2 & 0 & 0.2 \\ 0.7 & 0 & 0.1 & 0.2 \\ 0.4 & 0.4 & 0.2 & 0 \end{pmatrix}$$

The general model Mm, for arbitrary order m, is defined by the following transition probabilities:

$$\pi(x_1 x_2 \dots x_m, y) =$$
$$\mathbb{P}\left\{X_i = y \mid X_{i-m} = x_1, X_{i-m+1} = x_2, \dots, X_{i-1} = x_m\right\}$$

In our approach, we have previously extracted frequent motifs from the time series database. Figure 5.5 shows a motif that has been extracted from the database (above) and the transition probabilities for that motif (below), for the several orders from M1 up to M6. We follow the approach described in (Robin et al., 2007) and (Robin et al., 2005), where Markov chains are used to obtain expected counts of DNA motifs. Namely, we use Markov Chain models as the reference model to calculate the (estimated) expected probability μ of a motif to occur in the database. The probability is calculated with respect to transition probabilities. As they are a priori unknown, they are estimated according to the observed sequence. In fact,

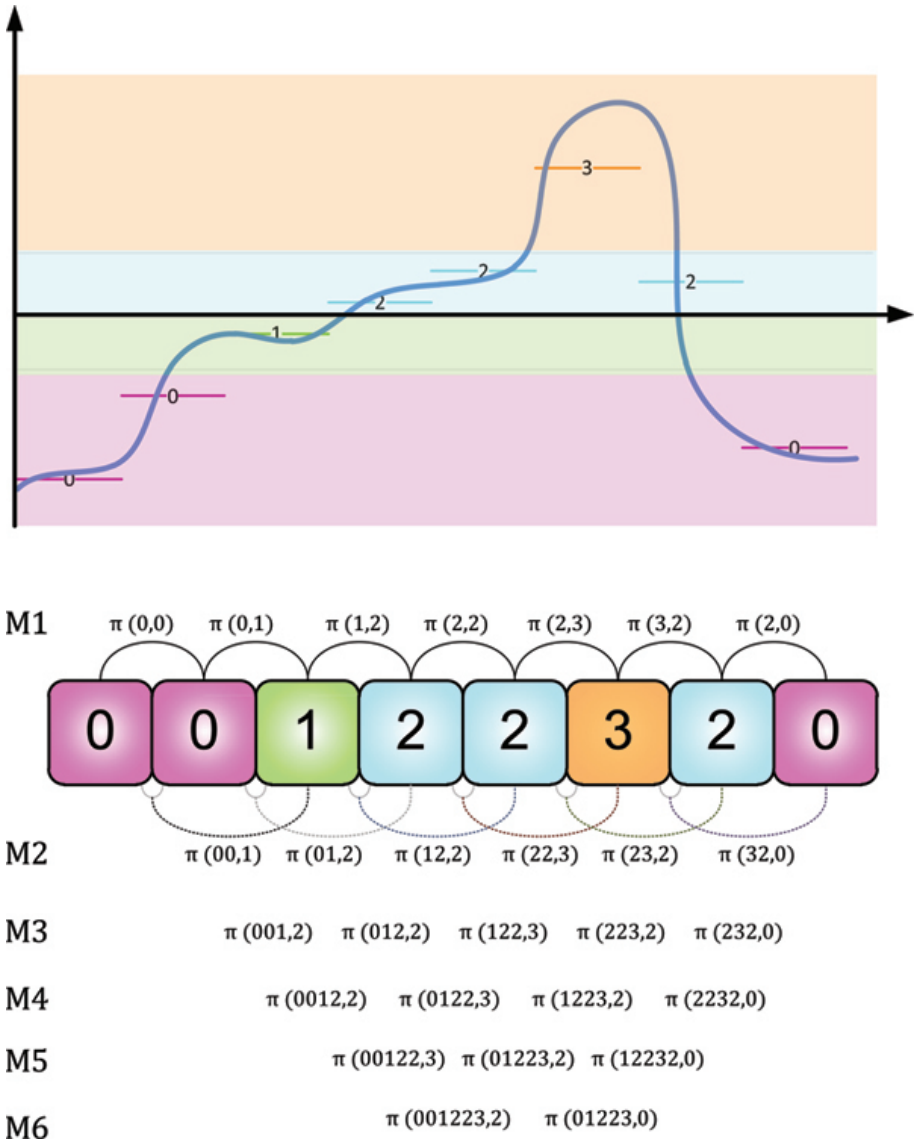


Figure 5.5: The transition probabilities for orders M1 to M6 (below) of the {00122320} motif (above).

they are estimated according to the corresponding observed counts $N(xy)$, i.e. their support. For example, under M1:

$$\pi(x, y) = \frac{N(xy)}{N(x+)}$$

where $N(xy)$ is the number of times y follows x in the database, and $N(x+)$ the number of times x occurs in the database, followed by any other symbol. In practice, we replace $N(x+)$ with the close approximate $N(x)$. For the general model Mm, we

have:

$$\pi(x_1 x_2 \dots x_m, y) = \frac{N(x_1 x_2 \dots x_m y)}{N(x_1 x_2 \dots x_m +)}$$

The observed counts contain all the information necessary to estimate the transition probabilities. The probability μ of a word $w = w_1 w_2 \dots w_l$ can be estimated by the product of the transition probabilities. For M1, we have:

$$\mu(w) = \mu(w_1) \times \pi(w_1, w_2) \times \dots \times \pi(w_{l-1}, w_l)$$

Generalizing to Mm:

$$\mu(w) = \mu(w_1 \dots w_m) \times \pi(w_1 \dots w_m, w_{m+1}) \times \\ \times \dots \times \pi(w_{l-m} \dots w_{l-1}, w_l)$$

As we can estimate transition probabilities using the observed counts for each subsequence, the probability of a word is calculated using the observed counts of its subwords of length m and $m+1$. For example, the probabilities μ for the motif $\{00122320\}$ of figure 5.5 are calculated as follows:

M0	$\mu = \frac{N(0)\,N(0)\,N(1)\,N(2)\,N(2)\,N(3)\,N(2)\,N(0)}{n_s^8}$
M1	$\mu = \frac{N(00)\,N(01)\,N(12)\,N(22)\,N(23)N(32)\,N(20)}{n_s\,N(0)\,N(1)\,N(2)\,N(2)\,N(3)\,N(2)}$
M2	$\mu = \frac{N(001)\,N(012)\,N(122)\,N(223)\,N(232)N(320)}{7n\,N(01)\,N(12)\,N(22)\,N(23)\,N(32)}$
M3	$\mu = \frac{N(0012)\,N(0122)\,N(1223)\,N(2232)\,N(2320)}{6n\,N(012)\,N(122)\,N(223)\,N(232)}$
M4	$\mu = \frac{N(00122)\,N(01223)\,N(12232)\,N(22320)}{5n\,N(0122)\,N(1223)\,N(2232)}$
M5	$\mu = \frac{N(001223)\,N(012232)\,N(122320)}{4n\,N(01223)\,N(12232)}$
M6	$\mu = \frac{N(0012232)\,N(0122320)}{3n\,N(012232)}$

where $N(x)$ is the count of motif x in the total (symbolic) length of the time series database n_s. We generalize the formulae for calculating μ in M0, M1 and the general order M(l-2). For a motif of length l, the maximal order is $l-2$.

M0	$\mu = \dfrac{\prod_{i=1}^{l} N(w_i)}{n_s^l}$
M1	$\mu = \dfrac{\prod_{i=1}^{l-1} N(w_i\, w_{i+1})}{n_s \prod_{j=2}^{l-1} N(w_i)}$
M($l-2$)	$\mu = \dfrac{N(w_1 \ldots w_{l-1})\, N(w_2 \ldots w_l)}{n\,(l-m+1)\, N(w_2 \ldots w_{l-1})}$

Under this scenario, the expected count of a motif is the product between the total number of words in the database (n) and the probability of the motif in the database:

$$\hat{N}_m(w) = n\,\mu$$

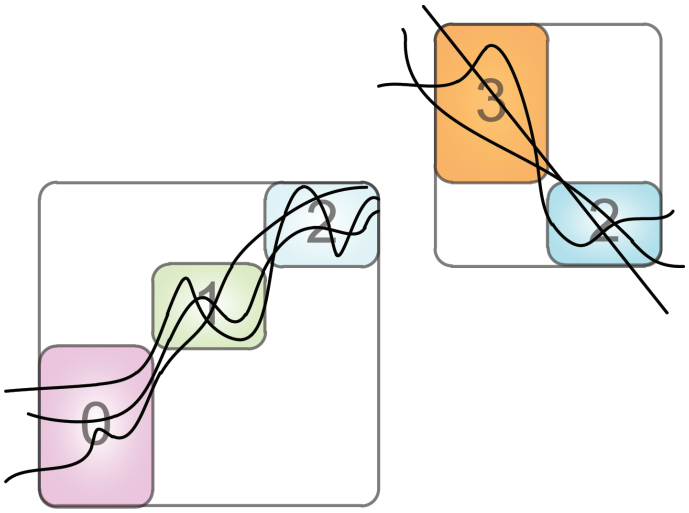


Figure 5.6: Example of subsequences of length 3 and 2 that compose M2.

Model Influence

The order of the model (m) is selected according to the length of the subwords composition we are interested, as it is known that Mm depends on its subwords of length $m+1$ and m. For example, in M2 we consider random sequences having on average the same composition as the observed sequence (regarding their subsequences of length 3 and 2). Figure 5.6 shows an example of subsequences of length 3 and 2 that compose M2 and possible instances that match submotif {012} (left) and {32} (right). Using a resolution (alphabet size) higher than 4 narrows the size of each block and allows to match more similar motifs (Castro and Azevedo, 2010).

The larger the order of the model, the larger the subsequences that are selected to compose the model. For that reason, larger models capture more information and mimic better the occurrences of a given motif. However, they may miss important information contained in the smaller subsequences (e.g. of length 2 and 3). A more suitable solution could be to consider all the orders and study how a motif's significance varies along the increasing order of the model(Robin et al., 2005). For instance, the high expected frequency presented by a motif can be caused by a very frequent subsequence of length 1. However, in this work we focus on a single model at a time. The model is essential to determine whether a motif is exceptional or not (Robin et al., 2005).

5.4.3 Assessing Statistical Significance

The expected counts have been estimated by a probabilistic model (Markov chains). However, expected counts by themselves do not provide enough information regarding the significance of motifs. Statistical hypothesis tests are widely used to help in decision making. In this setting, a null hypothesis is defined and then it is tested whether there is enough evidence in the data to reject that hypothesis. In motif discovery, the null hypothesis means that the given pattern is spurious or uninteresting, i.e. the actual motif count is similar to the expected one. It

means that if the motif count happens to be greater than expected, given that motif composition, it is so solely by chance. The null hypothesis is rejected in favor of the alternative hypothesis. In our case, that the motif has a frequency significantly greater than the expected count. We declare this motif statistically significant. After the hypothesis definition, it is necessary to define a test statistic and characterize its distribution. Our subject of interest is the motif count. Motifs counts distribution in the observed time series can be characterized as follows. Let the motif observed count w be:

$$N(w) = \sum_{i=1}^{n} Y_i$$

where Y_i is the Bernoulli random variable:

$$Y_i = \begin{cases} 1 & \text{if } w \text{ occurs in position } i \text{ in database } D \\ 0 & \text{otherwise} \end{cases}$$

with probability $p(Y_i) = \mu$. The motif count $N(w)$ is a sum of Bernoulli random variables. Therefore it follows a Binomial distribution:

$$N(w) \sim \mathcal{B}(n, \mu)$$

Note that the possible dependence between the different motifs is not an issue in our approach. Each motif count is treated independently of the others. However, we assume each instance (occurrence) of a motif is independent of one another. We can not guarantee that this assumption holds, due to the internal dynamics of the process that generated the time series at hand. Motif statistical significance is assessed by means of the *p-value*: the probability of the test statistic to present the observed value or a more extreme one, if the null hypothesis is true. That is to say, given the distribution for test statistic (the motif count), the p-value is the probability of the motif count to be at least as large as the observed motif count, just

by chance. It can be calculated by the probability of the $\mathcal{B}(n, \mu)$ random variable to be at least as large as $N(w)$. It is calculated by the complement of the Binomial cumulative density function, as follows:

$$\mathbb{P}(\mathcal{B}(n, \mu) \geq N^{obs}(w)) = 1 - \sum_{k=0}^{N(w)-1} \binom{n}{k} \mu^k (1-\mu)^{n-k}$$

The p-value is then compared to a predefined critical value (α). If it is no greater than α, the null hypothesis is rejected and the pattern is accepted as significant. In the literature, the critical value is typically set to 0.05. However, not considering the multiple hypothesis problem and fixing a value as the significance level tends to increase the false discovery rate (Holm, 1979). We use the Holm adjusted significance level (α') to control the number of false discoveries in the entire time series. This topic will be discussed in detail in section 5.4.5.

Besides the use of p-values to accept motifs that are statistically significant, they can also be used to sort the motifs of a given time series. This permits to achieve a rank of motifs according to their significance. If a p-value is very small, the motif is significantly frequent (over-represented).

5.4.4 Approximating p-values

To calculate p-values using the exact Binomial cumulative density function can be a computationally expensive operation, if n and k are large. This is the case in massive real-world data. Further, one should consider that the test must be executed for all extracted motifs. Approximate or asymptotic distributions are widely used in the literature (Marschall and Rahmann, 2009; Milo et al., 2002; Picard et al., 2008; Régnier and Vandenbogaert, 2006; Robin and Schbath, 2001; Robin et al., 2007), as they can reduce the computation time by one order or magnitude (see section 5.5). This difference stretches out along the size of the Binomial parameters. Typically, it is better to compute a computationally lighter analytic expression. They theoretically converge to the correct value as the sample size tends to infinity.

The Poisson approximation has been shown to fit correctly observed counts of words (Robin et al., 2007). Assuming this approximation, the motif count has mean and variance λ, i.e.

$$N(w) \sim \mathcal{P}(\lambda), \text{ with } \lambda = n\mu$$

The p-value is approximated by the tail distribution of the Poisson distribution:

$$\mathbb{P}(\mathcal{P}(\lambda) \geq N^{obs}(w)) = 1 - e^{-\lambda} \sum_{i=0}^{N(w)-1} \frac{\lambda^i}{i!}$$

The Gaussian approximation has also been used to approximate motif counts in bioinformatics. In this distribution, the motif count has mean $n\mu$ and variance $n\mu(1-\mu)$. That is,

$$N(w) \sim \mathcal{N}(n\mu, n\mu(1-\mu))$$

The p-value can be approximated by the following expression:

$$\mathbb{P}(\mathcal{N}(\mu, \sigma^2) \geq N^{obs}(w)) = 1 - \frac{1}{2}\left[1 + \frac{erf\Big(N(w) - 1 - \mu\Big)}{\sqrt{2\sigma^2}}\right]$$

where $erf(x)$ is the Gauss error function and is calculated as follows:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

The quality of the described approximations is experimentally analysed in section 5.5.

5.4.5 Controlling the risk of false discoveries

In classical hypothesis testing, the p-value is compared to a fixed α significance level, such as 0.05 or 0.01. In mining for statistical significant motifs we apply a test for each discovered motif, i.e. the number of tests applied is the number of distinct

motifs N_d. If α is set to 0.05 and we apply 100000 simultaneous tests to motifs that follow the null hypothesis, one would expect to find 5000 significant motifs by chance alone (Hanhijärvi et al., 2009). The larger the number of executed tests, the higher the chance to find at least one that incorrectly rejects the null hypothesis. This issue is known as the multiple hypothesis testing problem (MHTP) and occurs when multiple statistic hypothesis tests are performed simultaneously (Hanhijärvi et al., 2009; Webb, 2007). This will cause some motifs to be erroneously declared significant, i.e. false discoveries derivation. Traditionally, to control the number of false discoveries the significance level is set to values stricter than 0.05 or 0.01, to avoid an abundance of false positives (Storey and Tibshirani, 2003).

The Bonferroni adjustment (Hanhijärvi et al., 2009) is the classical and most simple approach to tackle the problem. The approach adjusts α to $\alpha' = \alpha/d$, where d is the number of hypothesis tests performed. However this value tends to be extremely strict (Hanhijärvi et al., 2009; Webb, 2007). An alternative method is the Holm procedure (Holm, 1979). This method provides a more reasonable α' level, while still maintaining the experimentwise significance level to α. The adjusted significance level is calculated as follows: all p-values are sorted increasingly from the smallest p_1 until p_d. For all $1 \leq j \leq d$, α' is set to the maximum p-value p_j that rejects $p_j \leq \alpha/(d - j + 1)$ (Webb, 2007).

The aim of both the Bonferroni and Holm procedures is to control the probability of at least one false positive, i.e. controlling the risk of committing even one type I error across the entire family of hypothesis tests - familywise error rate (FWER) (Hanhijärvi et al., 2009). This type of strong control is necessary when even a single false positive would be disastrous. For example, the hypothesis that several drugs are safe to use (Hanhijärvi et al., 2009), or the prediction of an earthquake leading to a city's evacuation. However, most of the time the FWER is too conservative (Storey and Tibshirani, 2003), as guarding against any single false positive occurring will lead to missed findings (false negatives - type II error). That is to say, the strong control is achieved at the expense of true motifs that are missed. In some applications, to enforce not having even one false motif is not necessary and too strict

(Benjamini and Hochberg, 1995). In particular, in problems where the number of hypothesis tested is very large, the probability to make any discovery at all becomes very small (Benjamini and Leshno, 2005). However, it may prove more suitable for applications where the number of significant motifs is extremely large. In some other applications, where it may be more appropriate to identify hypothesis for further study (exploratory analysis), it can end up being too strict.

We are in the presence of two extremes: the abundance of false positives caused by ignoring the multiple hypothesis problem, and the too strict control of FWER approaches leading to potential important motifs being missed. To bridge these two extremes the False Discovery Rate (FDR) was introduced (Benjamini and Hochberg, 1995). It is the expectation of the proportion of rejected true null hypothesis, among the rejected hypothesis (Benjamini and Leshno, 2005). That is, the expected proportion of falsely rejected hypothesis. It aims to describe the number of erroneous rejections, rather than whether any single error was committed (Benjamini and Hochberg, 1995). Formally, we test simultaneously d hypothesis (motifs), of which d_0 are true. Table 5.1 summarizes the approach outcomes:

Table 5.1: Outcomes of the motif statistical significance approach.

	declared non-significant	declared significant	total
spurious motifs	U	V	d_0
true motifs	T	S	$d - d_0$
	$d - R$	R	d

where R is an observable count and U, V, S, T are unknown. The FDR can be defined by dividing the number of false positives by the number of significant motifs output by the approach: the random variable $Q = V/(V + S) = V/R$ - the proportion of rejected null hypothesis erroneously rejected (Benjamini and Hochberg, 1995). The FDR is different from the false positive rate. Whereas the former is the rate that spurious motifs will be declared significant $V/(V + U)$, the latter is the rate that motifs declared significant are spurious. A false positive rate of 5% means that, on average, 5% of the spurious motifs in the database will be called significant. An FDR

of 5% means that, on average, 5% of the motifs called significant are spurious (Storey and Tibshirani, 2003). For that reason, it provides a sensible balance between true and false positives (S and V).

The FDR can be controlled at level q using the Linear step up procedure (Benjamini and Hochberg, 1995; Benjamini and Leshno, 2005). Similarly to the Holm procedure, we sort all p-values increasingly from the smallest p_1 until p_d. Let

$$k = max\left\{i : P_i \leq \frac{iq}{d}\right\}$$

then k motifs associated with $P_1, \ldots, P_k$ are significant. That is, we compare sequentially the ordered p-values to the constants linearly interpolated between q and q/m (Benjamini and Leshno, 2005), and keep rejecting the null hypothesis (declaring motifs as significant) until one fails to reject $P_i \leq iq/d$. This procedure controls the FDR at level q (typically set to 0.05) and aims to maximize the number of significant motifs for this level. That is, it assumes one should expect a given number of false discoveries and focuses on estimating what the FDR actually is (Zhang et al., 2004). Thus, more significant motifs will be derived by this approach. However, it allows the user to specify what percentage of the discovered motifs are spurious. In this work we compare the Holm and FDR approaches.

5.5 Experimental Analysis

In this section we describe the experiments performed using the proposed approach to analyze the statistical significance of time series motifs. First, the experimental methodology is outlined. Second, the datasets and their sources are described. Then, our approach is applied to datasets from various application domains and results are shown. The scalability of the approach is next analyzed. Finally, the quality of the Poisson and Gaussian approximations is evaluated according to existing measures.

5.5.1 Methodology

Motifs are extracted from the data using the MrMotif algorithm, with $K = \infty$, i.e. all patterns are extracted. See section 5.4.1 for the algorithm selection discussion. The iSAX *length* and *resolution* parameters are both set to 8, resulting in a $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$. The significance level (α) of the tests is automatically adjusted to cope with multiple testing. Instead of setting α to a typical value such as 0.05, we automatically derive the adjusted threshold using the Holm (Holm, 1979) and FDR (Benjamini and Hochberg, 1995) approaches. The Java implementation provided by MrMotif (Castro and Azevedo, 2010) authors is used. The Colt Library for High Performance Scientific and Technical Computing (v1.2.0) in Java is used for computing the Binomial, Poisson and Gaussian p-values. This library has been shown to provide accurate (long tail region) small p-values (Santosh Bangalore et al., 2009). The approach was implemented in the Java language and compiled using JDK 6. All experiments were executed in a machine with a Intel® Core^{TM} i5-530 processor with 4GB of RAM.

Our experimental methodology proceeded as follows. First, we extract frequent motifs from each of the presented datasets and calculate their statistical significance using the proposed approach. The number of statistical significant motifs is analyzed, considering the comparison between the Holm and FDR approaches to control the MHTP. A p-value is derived for each motif, assisting in the ranking of the different motifs. The scalability of the proposed approach is studied next. Then, the quality of the Poisson and Gaussian p-value approximations is compared, using several measures, to the Binomial Exact value. The aim of this work is not to provide proof of correctness for the statistical tests. Their theoretical properties are well established. Rather, we aim at showing their applicability and impact in the time series motif evaluation setting.

For clarity, we choose to use only one order for the Markov model from which we derive the motif expected probabilities. The chosen order is the maximal order (M6). We believe that this order is the most representative of the significance we are

interested in. However, calculations using smaller orders are also valid and should be used when the application at hand justifies it. The use of a model combining several orders remains as open research. Motifs of possible different sizes are accounted by treating each time series subsequence as a different time series (see section 5.3).

5.5.2 Datasets

We aim to test our approach on data from a wide range of applications and sizes. A set of 52 time series datasets available in the literature are selected from several sources. From (Yankov et al., 2007a), projectile shapes (*arrowhead*), brain activity (*eeg*) and motion-capture (*mocap*) data. Electrooculogram (*eog*) data from (Mueen and Keogh, 2010). Sensor networks monitoring (*sensorsnetwork*), telecommunication traffic (*telecom*) and protein data (*sasa*) from (Castro and Azevedo, 2010). Random walk data (*10*) from (Mueen et al., 2009b). Data from chlorine concentration measurements (*cl2*), Electrocardiogram (*koskiecg*), star light curves (*lightcurves*), graphical passwords (*pen*), exchange rate (*tickwise*) from (Shieh and Keogh, 2008). From (Keogh et al., 2005) we choose respiration (*nprs*), power demand (*powerdata*) and space shuttle data (*TEK*). Finally, datasets from a variety of sources are aggregated in (Keogh and Folias, 2002): airplane sensor data (*attas*), elastic burst (*burst*, *burstin*), chaotic time series (*chaotic*), sea level pressure (*darwin*), earthquake (*earthquake*), ECG (*ecg*), EEG heart rate (*eegheartrate*), brain imaging (*ERP*), fluid dynamics (*fluid*), Fortune 500 data (*fortune*), explosion sound (*infrasound*), laser measurements (*laser*), leaf images (*leaf*), electric signal (*leleccum*), logistic surrogate noisy data (*logistic*), fault detection (*mallat*), memory (*memory*), muscle activation (*muscle*), network (*network*), ocean depth (*ocean*, *oceanshear*), network packet delay (*packet*), power plant (*powerplant*), random walk (*random*), EEG (*rateeg*), image shape (*shapemixed*), standard and poor index (*sp*), speech recording (*speech*), stocks (*stock*), sunspots (*sunspot*), synthetic control charts (*synthetic*), and water level observations (*tide*) data.

5.5.3 Motif Statistical Significance Results

In this subsection the proposed approach is applied to the 52 different datasets generating more than 110000 distinct motifs. The statistically significant motifs returned by the approach are shown. The goals of the experimental analysis are: to show the pruning power of our approach, to highlight that it allows to avoid the use of unintuitive support of Top-K parameters as a pruning mechanism, to discuss whether p-value based motif ranking is an interesting approach, to compare the Holm and FDR approaches in our data and ultimately, to show the need for statistical tests in time series motifs mining. We first analyze the relation between sequence length (n), number of discovered motifs (N_d), number (NSM) and percentage (%) of significant motifs, and the cutoff level (α and α'). We show results for several datasets and false discoveries controlling approaches − the standard 0.05, Holm and FDR − in Table 5.2.

We can observe that larger datasets generate a larger number of frequent motifs. This is expected, since frequent motifs can be found even in random data. We can also see that a larger number of significant motifs are also extracted from larger datasets. In particular, when the MHTP is not considered ($\alpha = 0.05$), the number of significant motifs is large. Using the Holm procedure, however, there is no clear relation between dataset size and significant motifs in terms of percentage. The Holm approach prunes most of the false discoveries, since most of the motifs are not statistically significant. The percentage of accepted motifs is small for most of the datasets. Using the FDR to control false discoveries increases the number of discovered motifs, as expected, when compared to the Holm approach. The increase is large for some of the datasets. In some of the data, most of the frequent motifs clearly exceed their expectation (e.g. rateeg, ERP, cl2). In these cases, the FDR control is too permissive: it tends to converge to the standard 0.05 control. In those applications, to further reduce the FDR rate or using any FWER approach may prove a more suitable solution. Despite it appears that higher percentages of accepted motifs are present in larger datasets, some large datasets present a small

Table 5.2: Motif results for all datasets.

			0.05			Holm			FDR		
Dataset	n_s	N_d	NSM	α	%	$\downarrow NSM$	α'	%	NSM	**cutoff**	%
ERP	47616	2620	2612	0.05	99.69	757	2.684E-05	28.89	2612	4.987E-02	99.69
rateeg	576694	100438	70009	0.05	61.20	281	4.381E-07	0.25	45030	1.968E-02	39.36
lightcurves	5327	376	205	0.05	54.52	70	1.634E-04	18.62	154	2.061E-02	40.96
cl2	4310	54	44	0.05	81.48	36	2.778E-03	66.67	43	4.074E-02	79.63
koskiecg	2394	344	178	0.05	51.74	23	1.558E-04	6.69	106	1.555E-02	30.81
mallat	803	30	24	0.05	80.00	18	4.167E-03	60.00	22	3.833E-02	73.33
motor	420	60	25	0.05	41.67	8	9.615E-04	13.33	16	1.417E-02	26.67
stocks	18000	1402	532	0.05	37.95	7	3.584E-05	0.50	25	9.272E-04	1.78
arrowheads	1231	161	51	0.05	31.68	5	3.205E-04	3.11	13	4.348E-03	8.07
pen	510	46	17	0.05	36.96	4	1.190E-03	8.70	4	5.435E-03	8.70
sasa	81280	8146	5498	0.05	67.49	4	6.141E-06	0.05	403	2.480E-03	4.95
eog	67493	3101	1558	0.05	50.24	3	1.614E-05	0.10	34	5.643E-04	1.10
10	10000	754	259	0.05	34.35	2	6.649E-05	0.27	5	3.979E-04	0.66
powerdata	1838	291	128	0.05	43.99	2	1.730E-04	0.69	52	9.107E-03	17.87
shapemixed	160	14	6	0.05	42.86	2	4.167E-03	14.29	2	1.071E-02	14.29
TEK	180	50	19	0.05	38.00	2	1.042E-03	4.00	3	4.000E-03	6.00
burstin	1310	224	84	0.05	37.50	1	2.242E-04	0.45	18	4.241E-03	8.04
eegheartrate	373	85	38	0.05	44.71	1	5.952E-04	1.18	1	1.176E-03	1.18
insect	1471	73	20	0.05	27.40	1	6.944E-04	1.37	2	2.055E-03	2.74
leaf	442	72	29	0.05	40.28	1	7.042E-04	1.39	1	1.389E-03	1.39
network	1121	42	12	0.05	28.57	1	1.220E-03	2.38	1	2.381E-03	2.38
sensorsnetwork	555	9	1	0.05	11.11	1	6.250E-03	11.11	1	1.111E-02	11.11
attas	96	5	0	0.05	0	0	1.000E-02	0	0	1.000E-02	0
burst	488	4	0	0.05	0	0	1.250E-02	0	0	1.250E-02	0
chaotic	109	1	0	0.05	0	0	5.000E-02	0	0	5.000E-02	0
darwin	171	12	1	0.05	8.33	0	4.167E-03	0	0	4.167E-03	0
earthquake	209	6	0	0.05	0	0	8.333E-03	0	0	8.333E-03	0
ecg	56	8	2	0.05	25.00	0	6.250E-03	0	0	6.250E-03	0
eeg	62700	2880	1107	0.05	38.44	0	1.736E-05	0	0	1.736E-05	0
fluid	1662	17	1	0.05	5.88	0	2.941E-03	0	0	2.941E-03	0
fortune	500	9	0	0.05	0	0	5.556E-03	0	0	5.556E-03	0
infrasound	425	4	0	0.05	0	0	1.250E-02	0	0	1.250E-02	0
laser	194	29	11	0.05	37.93	0	1.724E-03	0	0	1.724E-03	0
leleccum	107	9	0	0.05	0	0	5.556E-03	0	0	5.556E-03	0
logistic	2000	181	64	0.05	35.36	0	2.762E-04	0	0	2.762E-04	0
lsf5	1496	7	1	0.05	14.29	0	7.143E-03	0	0	7.143E-03	0
memory	260	10	1	0.05	10.00	0	5.000E-03	0	0	5.000E-03	0
mocap	1370	70	18	0.05	25.71	0	7.143E-04	0	0	7.143E-04	0
muscle	188	27	7	0.05	25.93	0	1.852E-03	0	0	1.852E-03	0
nprs	1095	15	1	0.05	6.67	0	3.333E-03	0	0	3.333E-03	0
ocean	124	8	0	0.05	0	0	6.250E-03	0	0	6.250E-03	0
oceanshear	124	8	0	0.05	0	0	6.250E-03	0	0	6.250E-03	0
packet	2332	195	66	0.05	33.85	0	2.564E-04	0	0	2.564E-04	0
powerplant	154	5	1	0.05	20.00	0	1.000E-02	0	0	1.000E-02	0
random	1718	61	8	0.05	13.11	0	8.197E-04	0	0	8.197E-04	0
sp	921	28	3	0.05	10.71	0	1.786E-03	0	0	1.786E-03	0
speech	47	2	0	0.05	0	0	2.500E-02	0	0	2.500E-02	0
sunspot	146	3	0	0.05	0	0	1.667E-02	0	0	1.667E-02	0
synthetic	600	59	7	0.05	11.86	0	8.475E-04	0	0	8.475E-04	0
telecom	71	3	1	0.05	33.33	0	1.667E-02	0	0	1.667E-02	0
tick	903	16	1	0.05	6.25	0	3.125E-03	0	0	3.125E-03	0
tide	2906	9	0	0.05	0	0	5.556E-03	0	0	5.556E-03	0
TS	30	9	0	0.05	0	0	5.556E-03	0	0	5.556E-03	0

portion of accepted motifs. The relation between FDR accepted motifs and dataset size is not linear. Nevertheless, the results suggest larger datasets present a larger percentage of FDR controlled significant motifs.

The main conclusion to draw from these results is that the control of false discoveries is clearly required. Using the standard significance level at 0.05, without considering the MHTP, generates large percentages of significant motifs. It seems clear that most of these motifs are meaningless. The results suggest that the decision on which technique to control false discoveries is application dependent. In this work, we cover the Holm and FDR approaches. The user should select the approach taking into consideration several aspects. First, if the data will be further analyzed by the domain expert, we suggest the FDR approach. It outputs more statistically significant motifs, giving the opportunity to have more powerful interpretability. In the case of a prohibitively large number of significant motifs, the FDR rate should be made stricter (i.e. smaller than 0.05). To consider only the higher ranked motifs (in terms of significance) is also desirable. In case the motif mining approach output is not further inspected by experts, the Holm procedure seems the more suitable approach. It controls the probability of any single false positive. Second, the user should consider the weight of false positives vs. false negatives in the application at hand. If missing a significant motif is more problematic than erroneously declaring a motif significant, the FDR control should be selected.

For some datasets, all frequent motifs were discarded. Despite some of these data are large, no frequent motif could reject the null hypothesis. This indicates that using statistical tests in time series motif discovery can act as a filter, pruning meaningless motifs. This seems to support the need for statistical tests in time series motif discovery.

Pruning the prohibitively large output of pattern discovery algorithms is typically done by support or (Top) K parameters. These parameters are unintuitive and are typically optimized by experimentation. However, this is untenable in practice since the data are massive and it becomes very difficult to re-run the algorithms with a new parameter setting. Assessing motifs p-values avoids the use of unintuitive

parameters. We only consider the FWER or FDR levels to control false discoveries, and we set these to the standard values in the literature (0.05). The adjusted cutoff value is then automatically derived by our approach. In practice, no threshold setting is necessary to find the most statistically significant patterns in the dataset.

An interesting byproduct of our approach is that the motifs can be ranked according to their statistical significance, i.e. their p-value. Ranks organize the motifs in order of their evidence against the null hypothesis. Declaring a motif significant means that any other motif with more evidence against the null hypothesis should also be declared significant (Storey and Tibshirani, 2003). To be able to rank motifs is important, since a ranking yields a smooth way to select the patterns in the database that are most representative and relevant. The domain expert can further investigate those motifs for significance in the domain of study. In table 5.3 the highest ranked motifs for five of the datasets are presented. For simplicity, the numeric symbols are converted to alphabetic ones (respecting the alphabet index, i.e. $a = 0$ up to $h = 7$). Results for all datasets and full ranks (up to the least ranked motif) can be accessed in (Castro, 2011b).

It can be observed that motifs with the smallest p-value, i.e. highest ranking, present a large difference between their expected count and actual number of occurrences. The ranking produced by the approach is calculated using statistical tests, which are well established in the literature. Therefore, they reflect the degree of difference between expected and observed motif counts, which is the aim of the motif's p-value based ranking. Typically, the ground-truth motifs are not available in time series data, as the motif discovery process is unsupervised. To obtain a ground-truth about time series motifs can only be achieved by a domain expert, motif utility in a specific task (e.g. symbolic language) or interpretability (Minnen et al., 2007a). Even in the presence of a domain expert, some of the errors a motif discovery algorithm can incur are justified by real patterns that are simply unexpected (Minnen et al., 2007a). By introducing statistical tests in time series motif discovery, we intend to shed light on the motifs that are considered to present the highest statistical significance. As widely mentioned in the literature, statistical significance does not imply significance

Table 5.3: Most statistically significant motifs for several datasets

Datasets	**Motif**	$N(w)$	μ	**Expected**	**p-value**
sasa	gggfcbbb	17	3.9E-05	3.172479	4.77E-08
	hggdcbbb	8	8.79E-06	0.7143	8.93E-07
	bbbbgggg	14	3.37E-05	2.735099	1.19E-06
	bbbcggfg	10	1.67E-05	1.354194	1.68E-06
	abbdgggg	7	7.16E-06	0.58183	2.7E-06
eog	aacefggg	31	8.79E-05	5.932245	3.69E-13
	caacfggh	11	6.36E-06	0.429089	1.54E-12
	babbeggh	12	8.78E-06	0.592607	2.27E-12
	dbdgggfa	11	7.38E-06	0.497955	7.41E-12
	gabdeggd	12	1.03E-05	0.695669	1.2E-11
cl2	heddddbe	74	0.00193	8.319006	3.98E-13
	hecdccdf	37	0.001998	8.613394	7.54E-13
	hedcdccd	645	0.049903	215.0832	9.33E-13
	hedddcce	80	0.006069	26.1573	1.06E-12
	hedddccd	64	0.004855	20.92584	1.23E-12
koskiecg	gddddbg	40	0.002734	6.544641	2.37E-12
	dddddbfh	34	0.00299	7.157086	2.88E-12
	heddddb	43	0.006027	14.42812	7.89E-10
	dddddbgh	22	0.001817	4.350855	1.49E-09
	dbggdddd	45	0.00719	17.21198	1.55E-08
mallat	dgbcdche	90	0.03608	28.97219	6E-13
	cgbcdche	97	0.041707	33.49079	6.16E-13
	dgbbdche	92	0.038283	30.74089	6.57E-13
	dgbcdcge	59	0.024542	19.70757	7.29E-13
	dhbcdcge	137	0.056988	45.76165	7.92E-13

in a specific domain. However, to use the highest ranked motifs can provide a good starting point for the experts analysis. For example, the 5 highest ranked statistical significant motifs in protein unfolding data can provide the user a starting point to analyze the database for interesting motifs in that specific application. It is important that the expert considers only 5 motifs rather than 5000. In some cases, when the number of returned motifs makes the manual analysis very difficult, the use of p-value based rankings may become a requirement. We can also observe that motifs with the highest p-value also exhibit a large frequency. That is expected, since significant motifs are those whose frequency exceed their estimated frequency. There is no clear relation between motif count ranking and p-value ranking. However, some of the motifs with high frequencies are in the top p-value rankings, and vice-versa. Significant motifs are patterns whose actual frequency clearly exceeds their expected frequency. In this sense we can expect significant motifs to be frequent.

However, there are also significant motifs whose frequency is low. For example, if the expected frequency of a motif is 0.45 and the actual frequency is 2 it can be marked as significant. We also observe that some of higher ranked motifs are very similar to each other, for example in the *cl2* dataset. This may suggest a real bias in the data worthy of further investigation by the domain expert. It remains as open research to analyse the potential merge of these very similar motifs.

The MrMotif parameter setting does affect the extract motifs. For example, by slightly increasing the resolution and length of the SAX discretization, the number of possible motifs ($resolution^l$) increases significantly. This obviously impacts the significance computation. By having a much larger number of possible motifs, the probability of each individual motif becomes very low. This can lead to very low expected counts for motifs and inconsistent results for p-values. We follow the recommendation by SAX authors that 8 is a reasonable value for both SAX parameters and maintain these constant for all motifs, resulting in a consistent behavior. From our experience, this value is a good choice, as it provides a sensible balance. It guarantees a sufficiently large number of distinct motifs, but not too large to cause results degeneration.

The approach allows different length time series. However, all subsequences of a particular time series are limited to a fixed length. For different time series in the database the length can be different. As we are only interested in the shape of the series, if a series of length 1000 has a similar shape to a series of length 100 then they will match the same motif. In practice, this is achieved by seamlessly converting all series in the database to a SAX word of length 8. This enables different length series to match if they generate the same SAX word.

We show another practical example to highlight the relevance of the ranks generated by our approach. The most significant motif (showing the smallest p-value) from the *koskiecg* is displayed in Fig. 5.7. This motif is a well-known pattern in ECG data - the K-complex (Mueen et al., 2009b).

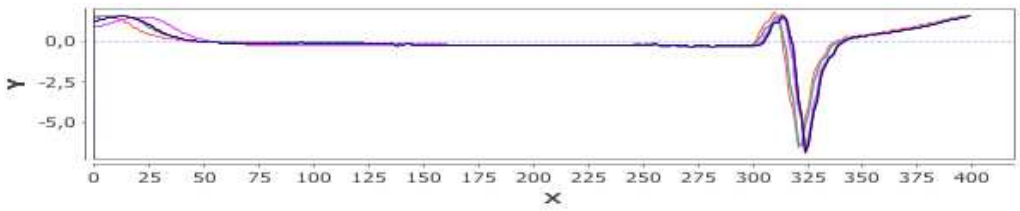


Figure 5.7: Motif with highest statistical significance in dataset *koskiecg*.

5.5.4 Scalability Experiments

In this section we study the execution time for the proposed approach. We use ten different sets of increasing size, ranging from 10000 to 100000 time series of length 1024, from (Mueen et al., 2009b). We use these data for two reasons: they have been used before and results on similar datasets are encouraged in order to walk towards data mining benchmarks (Castro and Azevedo, 2010); also, the size (in the Gigabytes) makes then attractive to test any approach. We apply our approach to each of the datasets. For each dataset, we record the motif extraction (MrMotif execution), and the statistical significance steps duration. We show the results in figure 5.8.

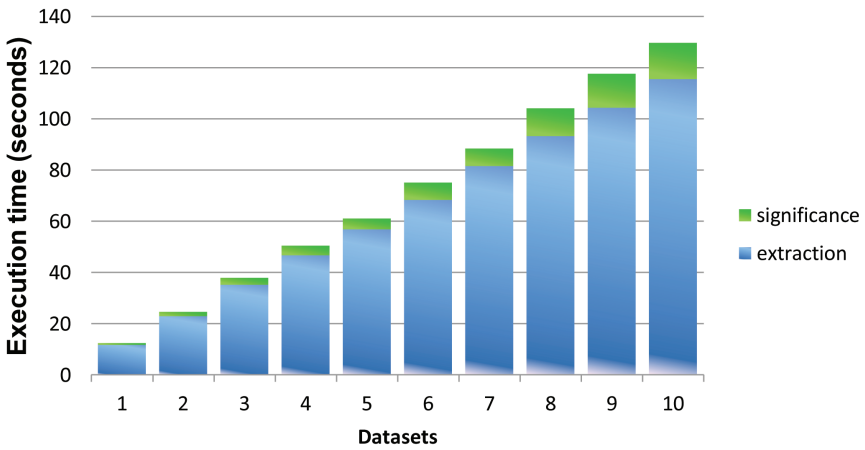


Figure 5.8: Execution time of the approach in ten increasingly sized datasets.

We observe that the approach increases linearly with the size of the time series dataset. The motif extraction step takes about 90% and the motif statistical significance analysis 10% of the total execution time, for all datasets. In figure 5.9 we zoom in

the execution time of the statistical significance part of our approach.

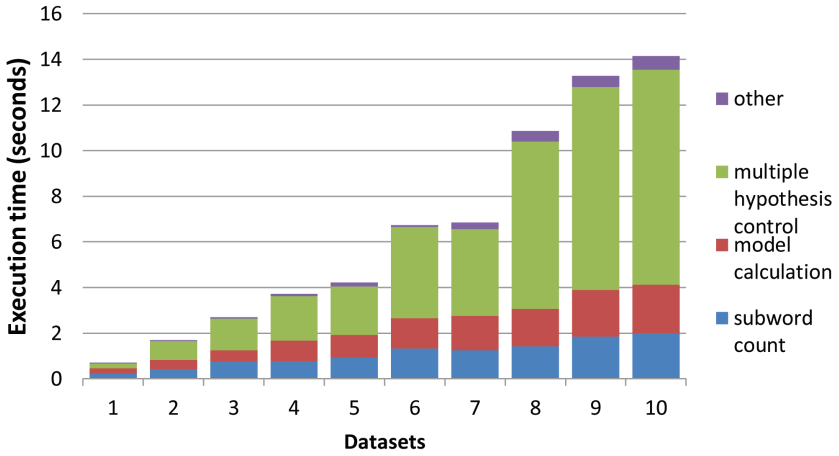


Figure 5.9: Execution time of the statistical significant part.

The larger portion of the approach is taken by the multiple hypothesis control substep (Holm and FDR). This substep sorts the p-values in order to find the appropriate adjusted cutoff value. The $O(n \log n)$ complexity of the sorting operation can be perceived. The other substeps of our approach seem to be negligible, considering we are only looking at 10% of the total execution time. Nevertheless, they increase linearly with the number of time series analysed.

To execute the motif extraction step (MrMotif) with $K = \infty$ does not decrease the performance of the approach. It always extracts and counts all motifs. The complexity of MrMotif is linear, as it only performs one sequential disk scan. Each motif is stored in a constant access time structure (hash table) and updated as the scan evolves. At the end of the scan we have the top-K motifs in main memory. In practice, to execute the algorithm with $K = 10$ or $K = \infty$ is equivalent.

5.5.5 Measuring the Poisson and Gaussian Approximations

The exact Binomial p-value calculation is computationally expensive for extremely large time series and motif counts. For example, with n=100000 and k=5000, the approximated p-value can be calculated about one order of magnitude faster than the exact one. It is therefore important to evaluate the quality of the p-value derived

by approximated approaches. In this work, two measures are used to quantify the agreement among the p-values produced by the different tests. The root mean square error (RMSE) is widely used to measure the difference between estimated values and actual values in prediction algorithms, for example. Hereby it is used to quantify the difference between the Binomial and the Poisson and Gaussian approximated p-values. It is calculated as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(E_i - O_i)^2}$$

where E_i is the Binomial test p-value for motif i, and O_i the approximation test's (Poisson or Gaussian) p-value.

The Total Variation Distance (d_{TV}) between the exact distribution p and its approximation $\hat{p}$, measures the greatest error one can make, in terms of probability, when using $\hat{p}$ instead of p:

$$d_{TV}(p, \hat{p}) = \sup_{A \subset \mathbb{N}} \mid p(A) - \hat{p}(A) \mid = \frac{1}{2}\sum_{n \geq 0} \mid p(n) - \hat{p}(n) \mid$$

In this subsection we calculate the RMSE and d_{TV} of the Binomial exact (B) and the Poisson approximation (P), for all datasets. Then, the same measures are applied to the Binomial and Gaussian approximation (G). The results for each measure are averaged for all datasets. In table 5.4 the average and standard deviation of the executed calculations are shown.

Table 5.4: RMSE and d_{TV} average and standard deviation

	RMSE(B.P)	dTV(B.P)	RMSE(B.G)	dTV(B.G)
Average	0.000193	0.002103	0.124324	24.6976
Std. Dev.	0.000251	0.00228	0.032015	105.1292

We can observe that the Poisson approximation is highly accurate, as both RMSE and d_{TV} present a very small average (and standard deviation), for all datasets. Therefore, it can be used as a replacement for the Binomial distribution. The

Gaussian approximation however, presents relatively large RMSE (average of about 12%) and d_{TV} values. These results support the experiments presented in (Régnier and Vandenbogaert, 2006), which has concluded that the Gaussian approximation is not suited to motifs.

To explore a possible relation between approximation quality and dataset size, the datasets are grouped in 4 groups of 13 datasets each and sorted according to their length. Results for the group RMSE average are shown in table 5.5.

Table 5.5: RMSE averages for each increasingly sized dataset interval.

N	Average RMSE(B.P)	Average RMSE(B.G)
1-180	0.000519	0.147843
188-600	0.000184	0.13305
803-1838	4.93E-05	0.121433
2000-576694	2E-05	0.094968

It can be observed that the RMSE and d_{TV} decrease as dataset increase in size, i.e. N grows larger, for both approximations. These results suggest that the approximation quality improves with dataset length. This result is somehow expected, since both Binomial and Gaussian approximations are asymptotic and are assumed to converge to the correct result as N grows to infinity.

We have studied the difference between exact and approximated p-values. It is also important to study whether p-values are under or over-estimated by each representation. To answer this question we have plotted all motifs for 9 of the datasets and their location in the chart with respect to the identity function ($f(x) = x$). Fig. 5.10 compares the Binomial and Poisson, and Fig. 5.11 the Binomial and Gaussian approximated p-values.

It can be observed that the Poisson and Binomial p-values are mostly situated on the identity function line. This is expected as results show that these two distributions yield very similar p-values (RMSE and d_{TV} comparison). The larger difference is between the Gaussian and Binomial results. It can be observed that most of the points in the scatterplot are above the identity function line. This means that the Gaussian approximation over-estimates p-values and by consequence under-estimates statistical significance.

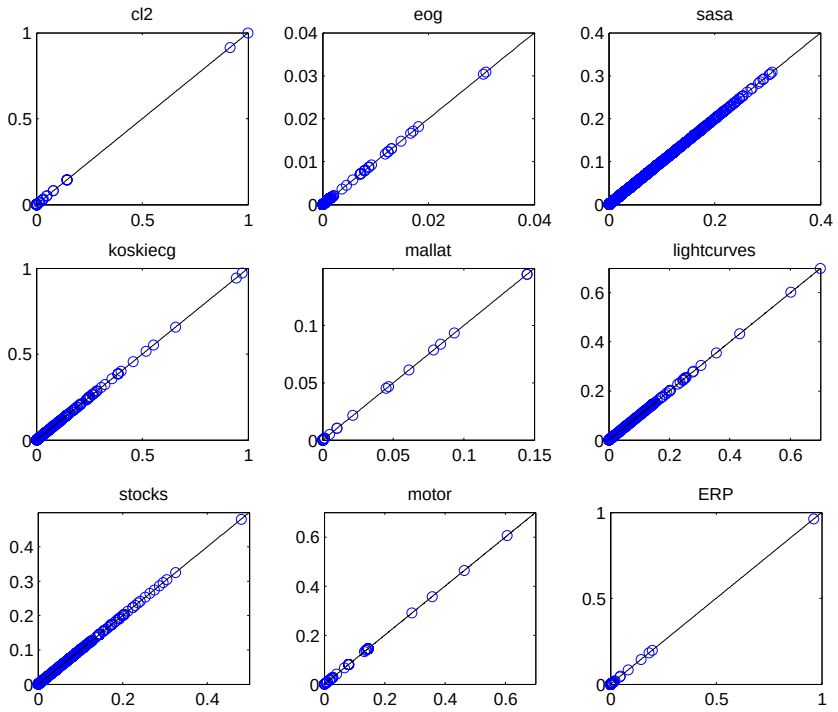


Figure 5.10: p-values of the Binomial (X axis) vs. p-values of the Poisson approximation (Y axis). The diagonal line is the graphical representation of the identity function.

5.6 Future work

Several directions to extend this work are possible. To study a Markov chain model using all the orders combined is currently underway. It may be interesting to analyze the impact of selecting another type of models, e.g. fractal models (Gao et al., 2004). To adapt our approach to the streaming time series data is another interesting possibility. It remains to be investigated whether this adaptation is trivial. Our approach enables further research on time series motifs extraction algorithms, reference models, motif evaluation measures and false discoveries controlling procedures. Any approach to tackle these tasks can be effectively plugged into our framework.

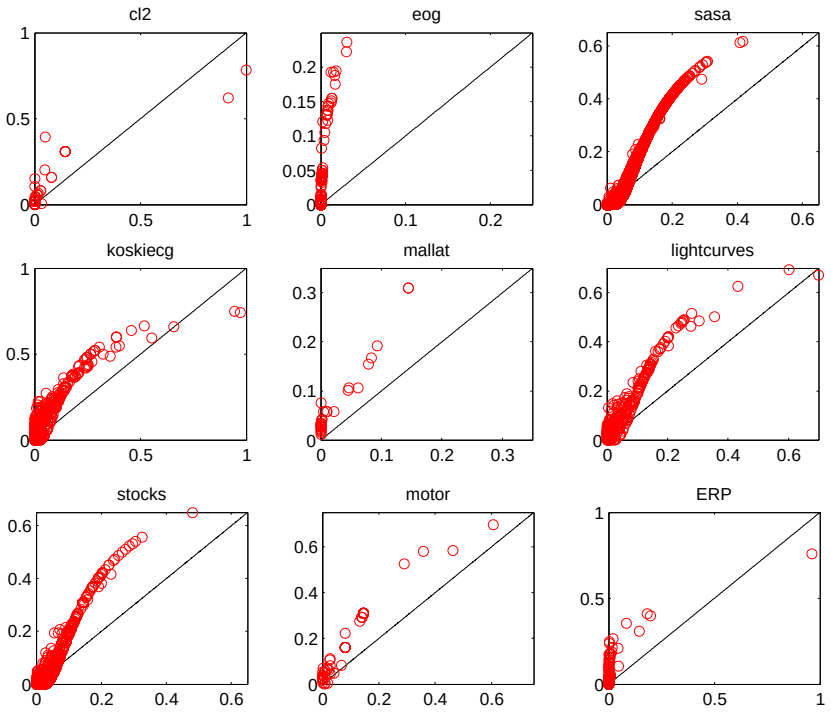


Figure 5.11: p-values of the Binomial (X axis) vs. p-values of the Gaussian approximation (Y axis).

5.7 Conclusion

We have proposed an approach to evaluate the significance of time series motifs using statistical significance tests. Our approach innovates by computing, for the first time in the literature, each time series motif p-value and declares a motif significant if its p-value is smaller than an automatically derived significance level. This circumvents the need to define unintuitive parameters like support or top-K in motif discovery algorithms. Further, it significantly reduces the number of returned patterns. An interesting byproduct is the ranking of motifs obtained by considering their statistical significance. We believe our approach provides researchers and practitioners with an important technique to evaluate the degree of relevance of each extracted motif. We also aim to highlight the importance of evaluating motifs since it is crucial to make motif mining an useful task in practice.

Reproducibility Note

All experiments, data and source code used are available online at (Castro, 2011b).

- 6 -

Automatically Estimating iSAX Parameters

6.1 Introduction

The large dimension of time series databases has generated the need for scalable algorithms to mine these data, including indexing (Agrawal et al., 1995; Camerra et al., 2010; Faloutsos et al., 1994), classification (Geurts, 2001), clustering (Kalpakis et al., 2001), anomaly detection (Dasgupta and Forrest, 1996), and motif discovery (Castro and Azevedo, 2010; Chiu et al., 2003; Mueen et al., 2009b). The key to the efficiency of these algorithms is to select a suitable representation of the data (Ding et al., 2008; Lin et al., 2003). Typically the data do not fit into main memory and disk I/O remains very inefficient (Lin et al., 2007), making the task of mining massive datasets difficult. Time series representations reduce the high dimensionality of the data, while maintaining its overall characteristics (Ding et al., 2008). The original time series is converted to an equivalent reduced representation, which fits in main memory. The problem is solved in main memory in the approximated space. Finally, only a few accesses to the raw disk resident data are required to confirm and tune the solution (Faloutsos et al., 1994). A plethora of time series representations exist, including the Discrete Fourier Transform (DFT) Faloutsos et al. (1994), the Discrete Wavelet Transform (DWT) (Chan and Fu, 1999), Piecewise Aggregate Approximation (PAA) (Keogh et al., 2001), Adaptive Piecewise Constant Approximation (APCA) (Chakrabarti et al., 2002), Single Value Decomposition (SVD) (Korn et al.,

1997), Symbolic Aggregate Approximation (SAX) (Lin et al., 2003, 2007) and its extension for indexing: indexable SAX (iSAX) (Shieh and Keogh, 2008). A brief review is provided in chapter 2. A comprehensive review is available at (Ding et al., 2008). From these, SAX has been widely used in the time series data mining community (Lin et al., 2007) as it is symbolic, reduces the dimensionality of the time series, allows lower bounding and is space efficient. The symbolic feature permits to take advantage of existing data structures and algorithms from the text mining and bioinformatics, specially tailored for symbolic strings. For example hashing, Markov models, suffix trees and decision trees (Lin et al., 2007). Lower bounding enables the definition of a distance measure in the representation space equivalent (lower bounding) to the Euclidean Distance (ED) in the raw time series space. This effectively makes the solution of data mining algorithms in the representation space to have identical results to the ones obtained from the original time series. Also, it enables the creation of time series indexes as the lower bound property is not available in other symbolic representations. Finally, it has equivalent performance to more complex approaches as DFT and DWT (e.g. in indexing) while using the same space (Lin et al., 2007). In practice, it permits to encode more information (a higher resolution) in the same number of bits. The SAX representation converts a time series of length n to a symbolic sequence of l symbols from an alphabet of size a. First, it divides the original series in l segments of size n/l. Then, it splits the amplitude of the series in a equiprobable intervals, according to the Gaussian curve, by using a set of breakpoints (6.1) Finally, each segment is averaged and assigned a symbol according to the interval the segment's average lies in. This process is shown in 6.2.

Setting the time series representation and abstraction level is a crucial step in data mining, as each dataset has its intrinsic dimensionality and cardinality (Hu et al., 2011). Finding the best parameter setting is currently open research and remains a "black art" (Hu et al., 2011). The need to set two parameters limits the applicability of the SAX representation, as the best parameter setting is highly dependent on the time series database. Typically, these parameters are set to a fixed value (e.g. 8), as

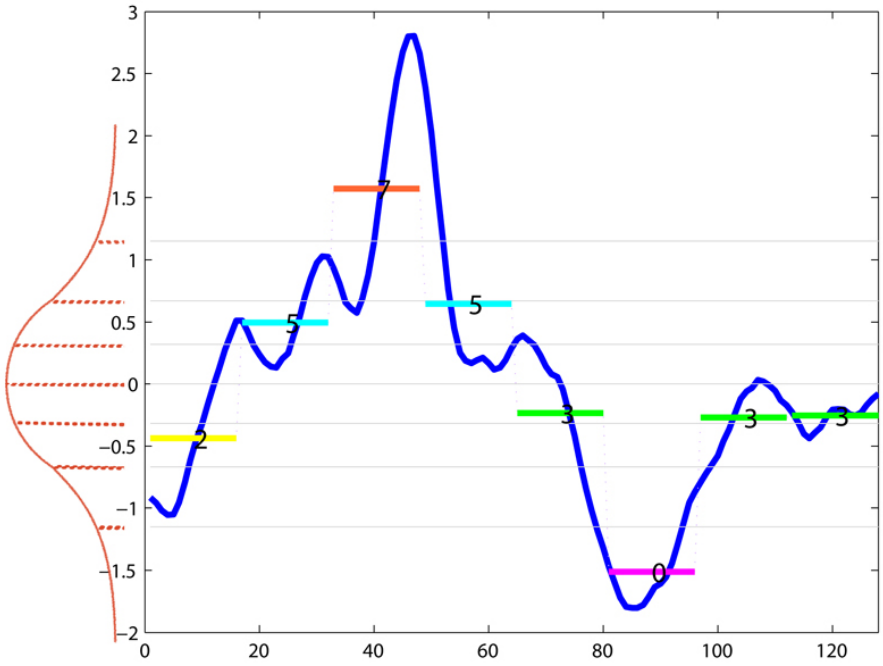


Figure 6.1: Conversion of a time series of length 128 to the length-8 SAX word $(2, 5, 7, 5, 3, 0, 3, 3)$

recommended by the authors (Ding et al., 2008; Lin et al., 2007) after experimenting with a large number of different datasets. However, it is straightforward to observe that the optimal adjustment of the number of segments (l) and the alphabet size of each segment (a) are specific to each time series in the database. Furthermore, even subsequences of the same time series can be better represented by different alphabet sizes. For example, the effect of using a fixed representation can be observed in Figure 6.2 (left) where a time series from the power demand dataset (available at Castro and Azevedo (2011)) is shown.

Observing only the representation (bottom) does not yield a good intuition on the original time series' shape. As only average information is encoded in the representation, two completely different series can generate the same representation, as long as they have the same mean. To resemble the original shape and consequently obtaining more accurate representations, it is necessary to also include the series' variability information. The right part of figure 6.2 will be referred later in this section after the approach is introduced.

The attentive reader can point out that in order to more accurately represent the

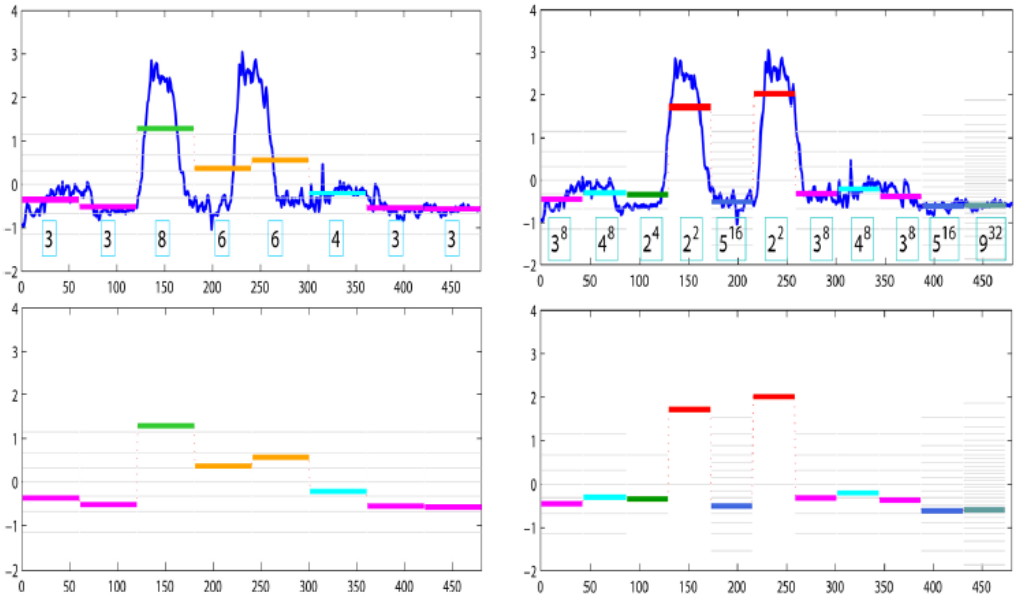


Figure 6.2: *Left*: Time series of five days' electric power consumption and iSAX with fixed $l = 8$ and $a = 8$ (top), generating the $(3, 3, 8, 6, 6, 4, 3, 3)$ word. Visual intuition of the blocks corresponding to the iSAX intervals for this length and resolution (bottom). *Right*: the same time series represented using AutoiSAX. The $(3^8, 4^8, 2^4, 2^2, 5^{16}, 2^2, 3^8, 4^8, 5^{16}, 9^{32})$ word of length 11 (top) is generated. Visual intuition of the blocks corresponding to the iSAX intervals for this length and resolution (bottom).

original time series, i.e. reduce the reconstruction error, larger symbolic lengths and alphabet sizes should be used. Despite this is generally true, it can lead to meaningless results in some data mining tasks. For example, in symbolic motif mining it can lead to no motifs being discovered. By selecting a larger alphabet, the amplitude space (block size) assigned to each symbol narrows considerably, making time series matching harder. In practice, to use alphabet sizes larger than 32 outputs only identical matches. If only identical time series match, motif discovery becomes meaningless. Although it is possible to mine motifs directly in the raw data, the existing approaches present quadratic complexity (Mueen et al., 2009b). Therefore, the use of efficient symbolic approaches is required in real-world data.

Using larger values for both parameters generates representations closer to the raw data. However, this does not solve the problem. In fact, to use too large symbolic length or alphabet sizes can lead to overfitting the data in classification or clustering tasks. In motif discovery, it can lead to no motifs being found. Hence, it leads to meaningless results. We illustrate this phenomenon with a simple example from

motif mining, where the aim is to find matching time series subsequences (i.e. that have the same SAX representation (Castro and Azevedo, 2010)). Figure 6.3 shows a length-8 motif extracted from the ECG database (6.5) using MrMotif algorithm (Castro and Azevedo, 2010). The blocks displayed are SAX symbols using an alphabet size of 4. In practice, a segment is assigned the symbol if its mean lies in that symbol's block. Two time series match the $2, 1, 3, 0, 1, 2, 2, 1$ motif. If the alphabet size is, for instance, increased from 4 to 8 the block size narrows considerably (to half the original size). It becomes increasingly harder for two different time series to match under those (much stricter) circumstances. Using an 8-alphabet size, the two time series generate two different SAX words: $4, 3, 7, 1, 2, 4, 4, 2$ and $4, 3, 7, 1, 2, 4, 4, 3$. They no longer match as their last segment differs slightly. If we generalize this example, it is trivial to observe that if too large alphabet sizes are selected, only identical time series can match. This leads to absence of matches, as the block size is too "fine-grained" for two non-identical subsequences to match. If only identical subsequences match, motif discovery becomes meaningless. We will further explore the issue in the experimental analysis (6.5).

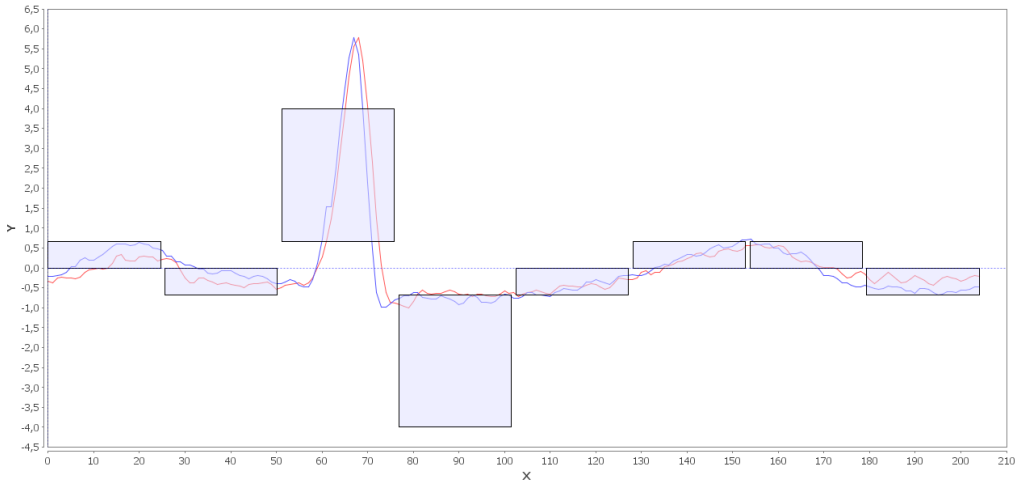


Figure 6.3: The $\{2, 1, 3, 0, 1, 2, 2, 1\}$ motif extracted from the ECG dataset

The current alternative to setting the parameters to a default value is to experimentally re-run the algorithm until the best parameter setting is found. In large datasets, to test all possible combinations of parameters is simply unfeasible (Castro and Azevedo, 2011). For example, to build an index with 1000000 time series can

take as much as 6 days (Camerra et al., 2010). Also, in unsupervised data mining tasks (e.g. clustering) it may not be possible to evaluate what the best parameter setting is, even by testing all possible configurations. Further, to use large alphabet sizes is not space-efficient. The only solution is to use a fixed parameter setting regardless of the dataset, which possibly leads to incorrect results. A method to automatically derive the best parameter setting "on-the-fly" is clearly necessary. In this work, we aim to tackle this limitation by introducing a novel technique to automatically derive the parameters of the iSAX representation. This technique, referred as AutoiSAX, not only discovers the best parameter setting for each time series in the database, but also finds the alphabet size for each iSAX symbol within the same word. Different sections of the series can behave differently (Figure 6.2). Using the same representation for all the symbols in the same series can lead to incorrect results. AutoiSAX is based on two very simple observations. First, more complex time series should have a larger symbolic length to better capture their complexity. Simpler time series (e.g. with little variation) can be correctly represented by fewer symbols. Second, time series sections presenting high variability with respect to their average need fewer symbols in order to capture this variability. Others, dispersing more closely to the mean value, require a larger alphabet size.

In Figure 6.2 (right), AutoiSAX is used to derive the representation's parameters. The generated symbolic length captures the series behavior much better than the fixed parameter. It also automatically generates a different alphabet size for each symbol, as they have very different behaviors. To fit bigger standard deviations' segments larger blocks are required (coarser resolution). For smaller deviation segments, smaller blocks suffice (finer resolution). Using AutoiSAX, it is straightforward to perceive the original time series shape by looking only at its representation (bottom). This is one of the primary goals of time series representations. We do not intend to show that larger parameters generate better representations. For example, in the automatic representation there are alphabet sizes and lengths smaller than the fixed ones. Our aim is solely to automatically determine the parameters, regardless of the relative order when compared to the fixed representation. In some cases the

generated parameters may be smaller or larger. The important aspect is that they are automatically generated.

We incorporate these observations into iSAX using two simple techniques. To capture a series complexity the Complexity Estimate (CE) (Batista et al., 2011) is used. It is computed by simply "stretching" the time series into a straight line and then measuring the line's length. It has been experimentally compared to twelve other more elaborate measures such as the Kolmogorov complexity and entropy with competitive results (Batista et al., 2011). Computing this measure requires low time and space complexity, zero parameters and is intuitive. For the variability calculation we use the simple standard deviation. The main contribution of this work is that one can automatically obtain the iSAX parameters in an efficient way. Currently, only brute-force techniques exist for the task. It also automatically derives, for the first time in the literature, different alphabet sizes within the same iSAX word. Previously, the use of different intra-word resolutions was limited to indexing, as it is untenable and cumbersome to manually define multiple resolutions for different symbols of the same iSAX word. In practice, our technique fully exploits iSAX capabilities to other mining tasks. Besides the parameter estimation is a major feature per-se, gains in visualization, classification, motif discovery performance are also obtained, as we will show in our experimental analysis. This section is organized as follows: in section 6.2 necessary background and notations are introduced. Section 6.3 analyzes existing SAX extensions and parameter estimation techniques. The AutoiSAX approach is presented in section 6.4. The experimental analysis of our approach is laid in section 6.5. In section 6.5.5 we discuss the main work breakthroughs and discuss future work.

6.2 Background and Notation

In this section we introduce some notations and useful definitions. First the object of study is defined.

Definition 6.1. A word $W = w_1 w_2 \dots w_l$ is the symbolic representation of a time series T, with $w_i \in$.

The symbolic length of word W is l. The $\sum$ is the representation alphabet. For example, $\sum = a, b, c, d$. The alphabet size is called the representation cardinality or resolution. The representation of time series S by a generic representation technique R is denoted by $R(T) = W$. For the scope of this work, the matching between two time series T_1 and T_2 can be defined as follows.

Definition 6.2. Time series T_1 and T_2 match if $R(T_1) = R(T_2)$.

In this work we are interested in the best representation technique available in the literature, as experimentally shown in (Ding et al., 2008) – SAX. For completeness, we also introduce SAX in this chapter. As a symbolic approximation, SAX takes two parameters l and a, and converts the raw time series T of length n into a sequence of l symbols (word) with an alphabet size a. This operation is represented by the following notation: $SAX(T, l, a)$, or simply $SAX(l, a)$, and operates as follows: First, the dimensionality of the time series is reduced by dividing it into l segments (word length) with the same length n/l using the Piecewise Aggregate Approximation (PAA) algorithm. This algorithm assigns to each segment its average value. Then, the amplitude of the time series is divided into a intervals, so that a symbol can be assigned to each interval. The best way to generate equiprobable intervals is to use $a - 1$ breakpoints that produce the same area under the Normal curve, as shown in Figure 6.2. These breakpoints can be obtained by using a statistical table and are shown in Table 6.1 for alphabet sizes 2, 4, 8, 16 and 32. Finally, symbols are obtained from the intervals. The segments below the smallest breakpoint are assigned the 0 symbol. The segments between the first and second breakpoints the symbol 1, and so forth. In order to assist calculations, binary numbers are used instead of actual symbols.

The iSAX representation extends classic SAX by allowing different alphabet sizes within the same word. To avoid ambiguity, the resolution of each symbol needs to be clearly stated in the iSAX word. For example, $W = 2^4, 0^8, 7^8, 9^{16}$. This enhancement

Table 6.1: SAX breakpoints for resolutions from 2 until 32

a	**2**	**4**	**8**	**16**	**32**
β_i					
β_1	**0.00**	-0.67	-1.15	-1.53	-1.87
β_2		**0.00**	-0.67	-1.15	-1.54
β_3		0.67	-0.32	-0.89	-1.32
β_4			**0.00**	-0.67	-1.16
β_5			0.32	-0.49	-1.01
β_6			0.67	-0.32	-0.89
β_7			1.15	-0.16	-0.78
β_8				**0.00**	-0.68
β_9				0.16	-0.58
...				...	...
β_{14}				1.15	-0.16
β_{15}				1.53	-0.08
β_{16}					**0.00**
β_{17}					0.07
β_{18}					0.15
...					...
β_{31}					1.86

enables the creation of a time series index. In practice, the use of this feature has been limited to indexing, as it is both cumbersome and untenable to manually set the resolution of each symbol. The problem is further aggravated by possibly having millions of iSAX words in the database. This is a central feature of our work. As we automatically calculate each symbol's alphabet size, our approach effectively enables to fully exploit iSAX capabilities in other tasks such as classification, motif discovery and visualization. For consistency, and because iSAX subsumes classical SAX, we will only refer iSAX for the remainder of the chapter. The exception will be section 6.3, as we refer works proposed before iSAX was introduced.

6.3 Related Work

The iSAX representation has been widely used in a large range of application domains and data mining tasks (Lin et al., 2007). Surprisingly, maybe due to the generality of the original approach, a very small number of works attempt to modify the actual representation technique (Lin et al., 2007). In Lkhagva et al. (2006), SAX is extended by adding two points to each symbol: the minimum and maximum points of that segment. The reasoning behind is that in some domains, e.g. financial data, the averaging process behind PAA can miss some important information. Adding maxima and minima information to segments yields further insight about the series that just the average simply cannot perceive. Preliminary results show the usefulness of the technique in some domains. Their goal is in principle similar to ours. We implicitly encode segment variability information in the representation. Rather than obtaining it through extreme points, we use the standard deviation. Also, we encode it in the original representation (accessing higher or lower resolutions), rather than using extra space along each symbol. This enables the user to capture more information about the raw time series in a space efficient way. Pham et al. (Pham et al., 2010) attempt to improve SAX by adding a pre-processing step where adaptive, rather than fixed (equiprobable) intervals, are discovered using clustering techniques. Experimental results show that it can outperform iSAX. In Yankov et al. (2007a) a motif discovery algorithm is proposed. The algorithm's parameter setting (including SAX's) is experimentally estimated in a smaller subset of training data. We aim to automatically derive the best parameter setting by using analytical tools, rather than using an iterative approach.To the best of our knowledge, there is no iSAX extension to automatically derive the parameters without experimentally probing the best configuration. An approach to find the best representation, cardinality and dimensionality of time series using the Minimum Description Length (MDL) principle has been introduced in Hu et al. (2011). This algorithm can be used to select the best parameter setting for a particular representation. In practice, it calculates the reduced description

length – the number of bits required to rebuild the raw time series – for every possible parameter configuration and selects the parameter configuration with the minimum length. The output configuration is the intrinsic cardinality and dimensionality of the time series. This approach is more generic than ours, as it can also detect the best model to use among the DFT, APCA, PLA representations, while also being parameter-free. AutoiSAX is specifically tailored to find the best parameter configuration of the iSAX representation. We analytically compute the best parameter configuration, rather than taking a brute-force approach, resulting into better computational complexity (section 6.5). Furthermore, our approach yields a specific parameter configuration for each symbol, rather than one single configuration for the entire (possibly long) time series. The per-symbol automatic configuration effectively enables to fully exploit iSAX intra-word multiresolution capability to other applications beyond indexing.

6.4 AutoiSAX

In this section we describe our approach to automatically discover the iSAX symbolic length and alphabet size parameters. Despite novel, the approach is straightforward to interpret as it is based on two very simple concepts. To derive the symbolic length parameter we focus on time series complexity. For the alphabet size we look into the time series variability. A complex time series can be intuitively defined as having more "peaks, valleys, and feature" than a simpler one (Batista et al., 2011). Our intuition is that a complex time series needs larger iSAX words, i.e. symbolic length, to better capture their complexity. On the other hand, simpler time series require smaller symbolic lengths. Complexity can be measured using many approaches, including information theory related (e.g. entropy), chaos based, Kolmogorov estimates, etc. In this work an alternative measure is used. The complexity estimate (CE) (Batista et al., 2011) is a straightforward measure of time series complexity. The idea is simply to "stretch" the time series into a straight line and measuring the line's length. It is based on the intuition that in a complex series,

the distance between every two consecutive points is larger than in a simpler time series. Is it computed by simply square rooting the sum of every two consecutive points' differences:

$$CE(T) = \sqrt{\sum_{i=1}^{n-1}(t_i - t_{i+1})^2}$$

It was empirically compared to several more elaborate measures with surprisingly competitive results (Batista et al., 2011). Furthermore, it has low space and time complexity, is intuitive and parameter-free. In order to maintain coherence between different length time series, the average per-point complexity estimate (ACE) is computed:

$$ACE(T) = CE(T)/|T|$$

This corresponds to step 1 in Algorithm 6.1 The smoothing effect caused by the average calculation of each time series segment in iSAX can miss some features of the raw data (Lkhagva et al., 2006). For example, extreme points can be absorbed by the mean. To tackle this issue, an interesting approach is to consider the variability of each segment's points regarding the segment's mean. That is, how far the points lie from the mean or how "spread" the points are with respect to the mean. To capture this intuition the simple standard deviation is selected. A segment with a large standard deviation needs a coarser interval to capture its variability towards the mean. Hence it requires a coarser resolution (alphabet size) to fit the original time series behavior. A segment with small standard deviation presents slight variation around the mean. This behavior "fits" in a narrower interval. The space-efficiency in our approach arises from using the actual iSAX representation resolution to encode the standard deviation, rather than requiring any extra storage. That is, each segment's standard deviation is encoded within the alphabet size parameter. The intuition is that segments having a larger standard deviation only need a few symbols (alphabet) to be able to capture the points' larger distribution. In order to represent segments with smaller standard deviations (more constant), a larger alphabet size is required. The AutoiSAX approach is presented in Algorithm

6.1. The actual parameter estimation consists of steps 2 (symbolic length) and 4 (alphabet size), which are explored separately in algorithms 2 and 3, respectively. For simplicity, the algorithm is hereby presented without any optimizations. First, the time series average complexity (ACE) is computed. The word length parameter is then calculated by framing the ACE between the minimum and maximum word length for the series, 2 and $(|T|)/2$, respectively. Then, a modified PAA algorithm is executed in order to output, besides each segment's mean, its standard deviation. Next, the standard deviation of each segment (pre-symbol) is converted to an alphabet size between the minimum (2) and maximum (32) resolution. In section 6.5 we show why resolutions larger than 32 are not used in our approach. Finally, each segment's mean is mapped to the corresponding symbol, according to the previously calculated symbol's alphabet size.

Algorithm 6.1 AutoiSAX(time series T)

1: $Compl \leftarrow ACE(T)$
2: $l \leftarrow complexityToLength(Compl)$
3: $paa\_stds \leftarrow PAA'(T, l)$
4: $resolutions \leftarrow stdevsToResolutions(paa\_stds)$
5: $W = mapToSymbols(PAA', resolutions)$
6: **return** W

The symbolic length parameter is estimated in step 2 by the *complexityToLength* function. It takes a complexity average and maps it into a symbolic length. As the time series is normalized with zero average and one standard deviation, the average complexity ranges between 0 and the maximum complexity. The maximum possible average complexity is experimentally determined to be 0.16. This function simply performs a linear interpolation of the series complexity into the symbolic length range between 2 and $(|T|)/2$, as shown in 6.2. The minimum ACE is set to 0.00001 to obtain a 5 digit precision in the calculations.

The *stdevsToResolutions* function (step 4) takes a list of segments' standard deviations and maps it to the final word list of resolutions. The mapping is based on a simple interpolation between the minimum (2) and maximum alphabet sizes. The maximum alphabet size is set to 32, as any further increase generates meaningless results in symbolic motif discovery. The intuition of this approach is based on a

Algorithm 6.2 complexityToLength(complexity c)

1: $c_0 \leftarrow 0.00001; c_1 \leftarrow maxComplexity$
2: $l_0 \leftarrow 2; l_1 \leftarrow |T|/2$
3: $l = \dfrac{(c - c_0) \times l_1}{c_1 - c_0} + l_0$
4: **return** $floor(l)$

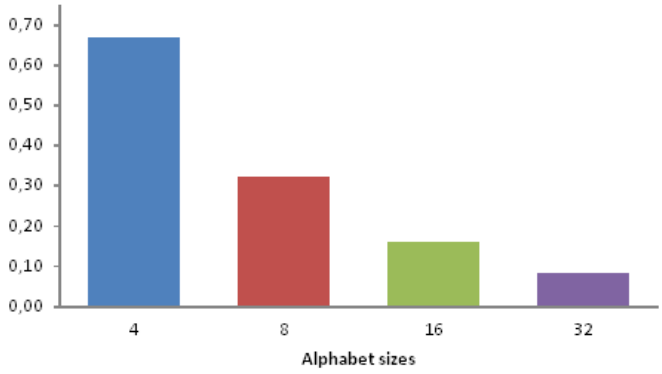


Figure 6.4: Size of the MRD intervals for several resolutions

simple idea taken from the breakpoints table 6.1. Let β_{zero} be the zero crossing breakpoint for a given alphabet size (shown in bold). The minimum resolution difference (MRD) for that resolution is $\beta_{zero} - \beta_{zero-1}$. For example, considering resolution 8, $\beta_{zero} = 4$ and MRD= 0.16. Table 6.2 shows the MRD for the resolutions 2 up to 32.

Table 6.2: β_{zero} and MRD for several resolutions

a	**2**	**4**	**8**	**16**	**32**
β_{zero}	β_1	β_2	β_4	β_8	β_{16}
MRD	∞	0.67	0.32	0.16	0.08

In practice, the MRD is a symbol's block size for a given resolution. That is, the minimum interval where a segment's mean can lie in order to be assigned a given symbol. Figure 6.4 shows the size of MRD intervals for the above resolutions.

The larger the resolution, the narrower is the interval. As the maximum standard deviation we aim to enclose corresponds to an alphabet size of 2, we can safely assume $maxStd = 2 \times MRD(4)$. Generally speaking, if the segment's standard deviation is

smaller than a resolution's minimum block size, then the symbol is assigned to that resolution. Thus, the linear interpolation technique seamlessly interleaves between the target resolutions, according to the obtained standard deviation (Algorithm 6.3).

Algorithm 6.3 stdevsToResolutions ($paa\_stds[]$)

1: **for** sd_i in paa_stds **do**
2: $s_0 \leftarrow 0; s_1 \leftarrow maxStd$
3: $a_0 \leftarrow 2; a_1 \leftarrow 32$
4: $a' = \frac{(sd_i - s_0) \times a_1}{s_1 - s_0} + a_0$
5: $a = (a_1 - a') + a_0$
6: $r_i = floor(a)$
7: **end for**
8: **return** r

The base of every time series data mining task is time series similarity (Ding et al., 2008), which is typically measured as dissimilarity in the form of distance measures. One of the main iSAX features is that it enables the definition of a distance measure ($mindist$) that lower bounds the Euclidean Distance in the original time series. In this work, two same-length time series can generate different length iSAX words. The iSAX mindist function is only defined to compare words with the same length. In order to enable AutoiSAX to be used in classification, motif discovery, etc., we slightly modify the original mindist function to enable the comparison of different length iSAX words. We show the modified mindist' function in Algorithm 6.4.

Algorithm 6.4 mindist' ($\hat{Q}, \hat{C}$)

1: let $m = min(|\hat{Q}|, |\hat{C}|), M = max(|\hat{Q}|, |\hat{C}|)$
2: $mindist'(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{M}} \times \sqrt{\sum_{i=1}^{M} (dist(\hat{q_j}, \hat{c_k}))^2}$
3: **for** sd_i in paa_stds **do**
4: $s_0 \leftarrow 0; s_1 \leftarrow maxStd$
5: $a_0 \leftarrow 2; a_1 \leftarrow 32$
6: $a' = \frac{(sd_i - s_0) \times a_1}{s_1 - s_0} + a_0$
7: $(a_1 - a') + a_0$
8: $r_i = floor(a)$
9: **end for**
10: **return** r

In practice, we use the raw time series to drive the alignment between the different sized representations. For example, when comparing $(3^8, 4^8, 2^4, 2^2, 5^{16}, 2^2)$ with $(3^{12}, 4^{16}, 5^{16})$ the following comparisons are performed: 3^8 and 4^8 compare to 3^{12}, 2^4 and 2^2 to 4^{16}, 5^{16} and 2^2 to 5^{16}, as every symbol in the smaller word will represent one or more symbols from the larger word. It is not required that the word sizes are multiple, as symbol j of one series is compared to its corresponding symbol k in the other series. Thus, the admissible lower bound provided in the original iSAX's *mindist* is preserved (Shieh and Keogh, 2008).

6.5 Experimental Analysis

In this section we perform the experimental analysis to show the validity and impact of our approach. The experimental analysis uses different benchmark datasets for execution time (6.5.2), classification (6.5.3) and motif mining (6.5.4). Each dataset is briefly described in the beginning of each subsection. All experiments were executed on a machine with an Intel® Core™$i5-530$ processor with 4GB of RAM. Unless explicitly stated otherwise, Java implementations compiled with JDK 6 were used. The experimental analysis proceeds as follows. 6.5.1 analyzes the danger of using too high resolutions. The computational performance of the approach is described in 6.5.2. In subsection 6.5.3 the impact of using AutoiSAX in classification is assessed. Finally, in 6.5.4 we apply our approach to motif mining.

6.5.1 The danger of selecting large resolutions

In section 6.1 a simple example from motif mining is shown to explain why one should not select too large resolutions. In this section, we aim to empirically demonstrate why this is the case. First, let us properly motivate the need for such experiment. We have observed that the alphabet size parameter is typically set to very large values. For example, tightness of lower bound (TLB) is widely used to compare representation techniques and their distance measures (Ding et al., 2008; Shieh

and Keogh, 2008). It provides a good approximation of indexing performance, as it is hardware and implementation independent (Esling, P. and Agon, C., 2011). Intuitively, it calculates how closely a representation's distance measure approximates the Euclidean Distance (ED). It is used for the comparison between different representation techniques (e.g. iSAX to DFT, etc.). In these experiments, the iSAX alphabet size parameter is often hard-coded to 256 (Ding et al., 2008; Shieh and Keogh, 2008). The authors argue that as iSAX is more space efficient, as it can afford to encode a higher resolution using the same space than, for example, DFT. Indeed this is true. However, setting the alphabet size to 256 different symbols in a normalized series with mean 0 and standard deviation of 1 is close to useless. The minimum interval size corresponding to a 256 resolution is of 0.01. We encourage the reader to re-read this value: 0.01. To use such an interval to distinguish between different iSAX symbols at such fine resolution is similar to using the raw data itself. We claim that using such resolutions is both unpractical and unfair to competing representations. It is expected that by using a resolution so close to the raw data one obtains a much tighter lower bound. It is also straightforward to observe, and has been shown 9 years ago (Lin et al., 2003), that in tasks where the degree of proximity to the raw data is important, higher resolutions will outperform lower ones.

To analyze the impact of resolutions in motif discovery, 52 datasets from a wide variety of sources, aggregated in Castro and Azevedo (2011), are selected. For each dataset, we interleave between the resolutions 2, 4, 8, 16, 32, and 64, and record the number of symbolic motifs extracted. The motif extraction algorithm is simply to generate an iSAX word for each time series received as input and counting the number of times each word appears. The MrMotif algorithm (Castro and Azevedo, 2010) is suitable for this task, as it gracefully interleaves between resolutions and outputs the symbolic motifs discovered for each resolutions. In Figure 6.5, the number of discovered motifs (N_d) for each resolution and dataset are shown. For space, 7 datasets are displayed. Results for the remainder of the datasets are similar. The names *eeg* and *EEG* refer to different datasets.

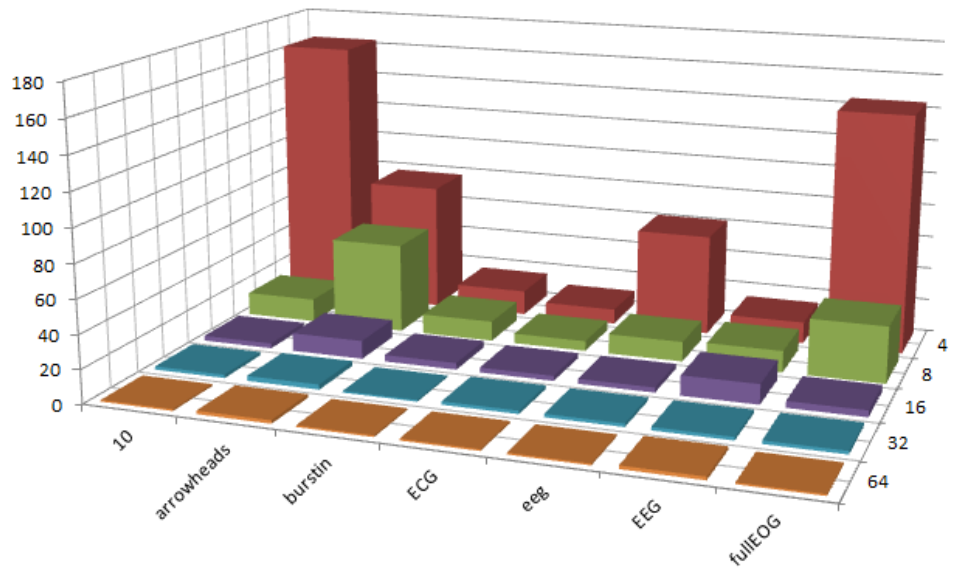


Figure 6.5: Number of symbolic motifs found for several resolutions and datasets.

It can be observed that N_d decreases as one increases the alphabet size. By looking closely, very few motifs can be found at resolution 32 or larger. In fact, at resolution 64, either no motifs or only identical motifs can be found. The larger the resolution, the finer are the intervals available for the time series' segments. This makes it hard in practice for two non-identical time series to match. This suggests that using resolutions larger than 32 in approximate motif discovery is of little value, and is therefore meaningless.

6.5.2 Computational Performance

In this subsection we study the proposal's execution time and compare it to the MDL based parameter estimation approach (Hu et al., 2011). The authors' MATLAB® code for the competing approach is used. For the sake of fair comparison, we use an AutoiSAX MATLAB® implementation. Ten different sets of increasing size from (Mueen et al., 2009b) are used, ranging from 10000 to 100000 time series of length 1024. We use these data for two reasons: they have been used before and results on similar datasets are encouraged; in addition, the size (in the gigabytes) makes them attractive to test any approach. All the time series in each set are concatenated

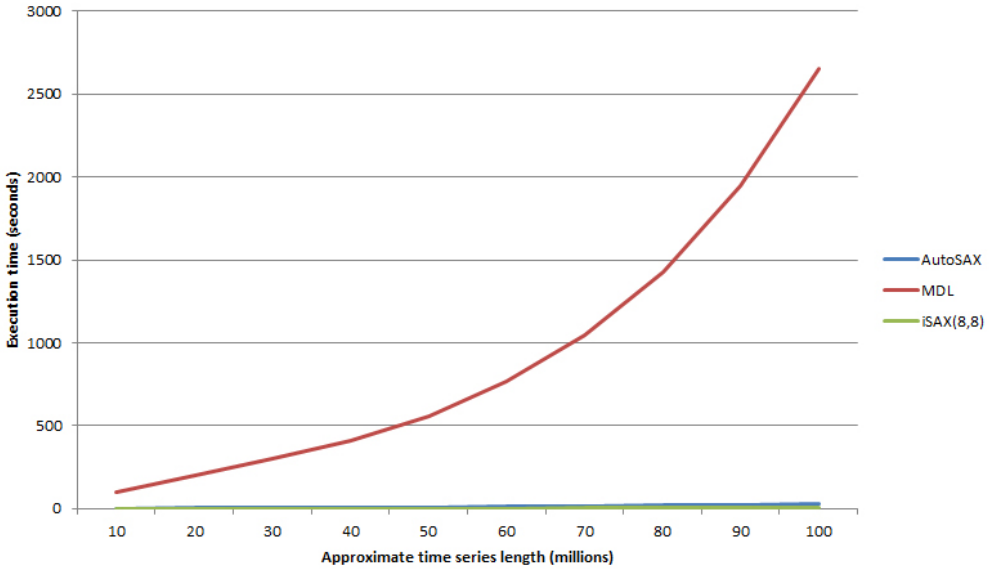


Figure 6.6: Execution time of AutoiSAX, fixed iSAX and MDL approaches in ten increasingly sized time series.

generating ten time series with lengths 10240000 up to 102400000. We apply both techniques to each time series ten times and record the average execution time for each series. Results are presented in Figure 6.6.

For the 10 million-sized time series, the techniques execution time is as follows. For iSAX: 0.40 seconds, AutoiSAX: 1.45 seconds, MDL: 102s. It can be observed that the MDL approach's execution time increases quadratically with the time series size. The AutoiSAX and iSAX methods increase linearly. The MDL is about two orders of magnitude slower than iSAX and about 70 times slower than AutoiSAX, on average. In practice, for estimating the parameters of a single time series taking 1s to convert to the AutoiSAX representation, MDL would take longer than one minute. As a brute-force approach it needs to scan all possible models from lengths 2 up to 64, and the (simplified) resolutions 2, 4, 8, 16, 32 and 64. To calculate one single model costs just slightly higher than AutoiSAX. However, the approach requires that all models are computed, in order to select the best. This effectively makes it perform 372 model calculations instead of just one. To further inspect the impact of MDL approach' execution time, we performed an experiment with a small dataset ("50words") from the UCR classification benchmark (Keogh et al.,

2009). The dataset consists of 450 training and 455 test series of length 270, and 50 classes. The MDL approach was selected for estimating the iSAX parameters for 1-NN classification using the iSAX's *mindist* distance measure. After one hour of execution, the approach had not yet terminated execution. As the approach's execution time makes it untenable to use in real word data mining tasks, we omit it from the remainder of the experimental analysis. The AutoiSAX presents linear complexity. However, the constant factor is larger than the fixed iSAX, as another scan over the original series is required for the complexity (ACE) and standard deviation computation.

6.5.3 Classification Accuracy

In this section the 1-NN classification experiment performed in the previous subsection is extended. The AutoiSAX and fixed iSAX (using several resolutions) are compared for all datasets in the UCR classification benchmark in terms of classification accuracy. One-NN if often used to argue the suitability of a distance measure or representation (Ding et al., 2008). Our aim is not to analyze the applicability of AutoiSAX for classification. It has been show that using 1-NN with the Euclidean Distance directly in the raw data is surprisingly competitive and fast. Rather, we aim to show that AutoiSAX can obtain results as accurate as fixed iSAX, without the hassle of manually optimizing the parameters. The benchmark is composed of 42 datasets with training sets containing 16 to 1000 and testing sets with 28 up to 3582 time series. Each time series length ranges from 24 to 1882, with 2 up to 50 classes. The MATLAB® classification framework provided in the benchmark is used. The single obvious modification is changing the distance measure for the selected representation. A scatter-plot of the AutoiSAX and fixed iSAX pairs of results for each dataset are displayed in Figure 6.7. Thus, one point represents the classification accuracy of the pair (AutoiSAX,iSAX) for one dataset. Points above the identity function (blue) mean that AutoiSAX outperforms fixed iSAX for that dataset and vice-versa.

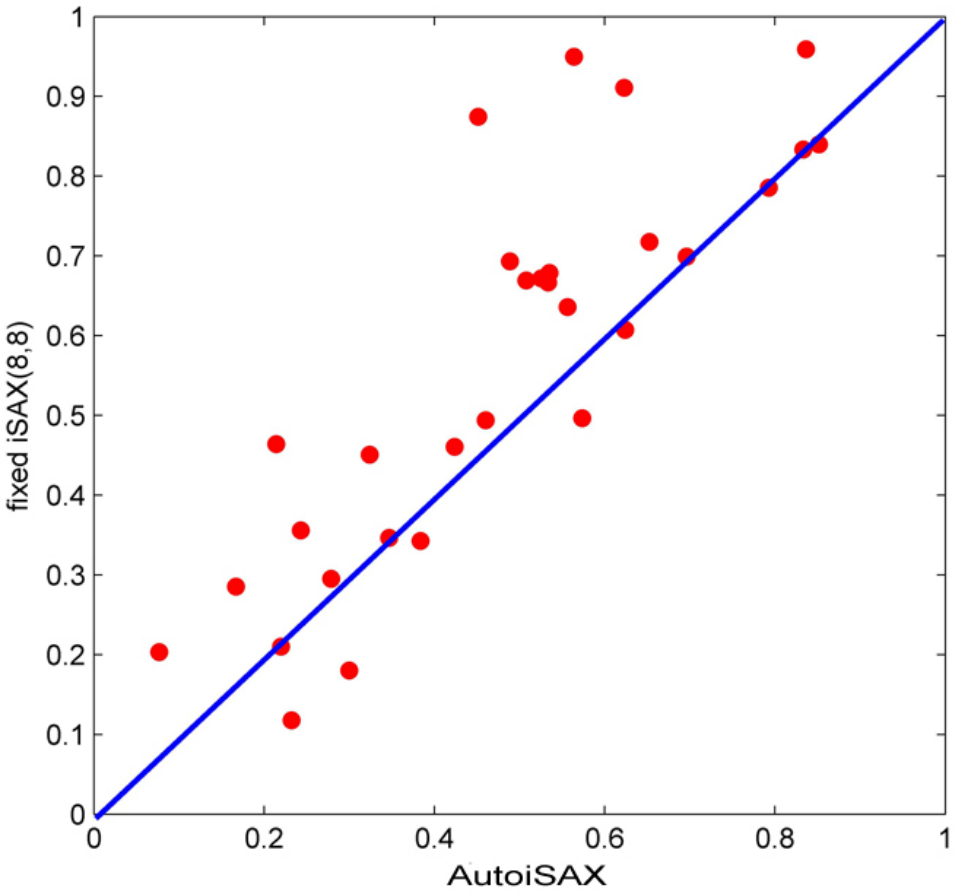


Figure 6.7: Accuracy of 1-NN classification on 42 datasets: auto SAX vs fixed iSAX $(8, 8)$. Above the line AutoiSAX outperforms fixed iSAX.

In general, the AutoiSAX approach showed better classification accuracy than the fixed one. For the remainder of the datasets, the approaches showed similar accuracy. This suggests that our approach approximates the raw data in a more realistic way, and this fact has repercussions in classification accuracy.

6.5.4 Motif Discovery

In this subsection we apply AutoiSAX to the task of motif discovery. We use the same simple algorithm as in 6.5.1: generating an AutoiSAX word for each time series and counting repetitions. We compare the approach to fixed iSAX with length and resolution 8, and with an average of fixed iSAX resolutions ranging from 2 up to 64. MrMotif is selected as it can seamlessly provide these results. The 52

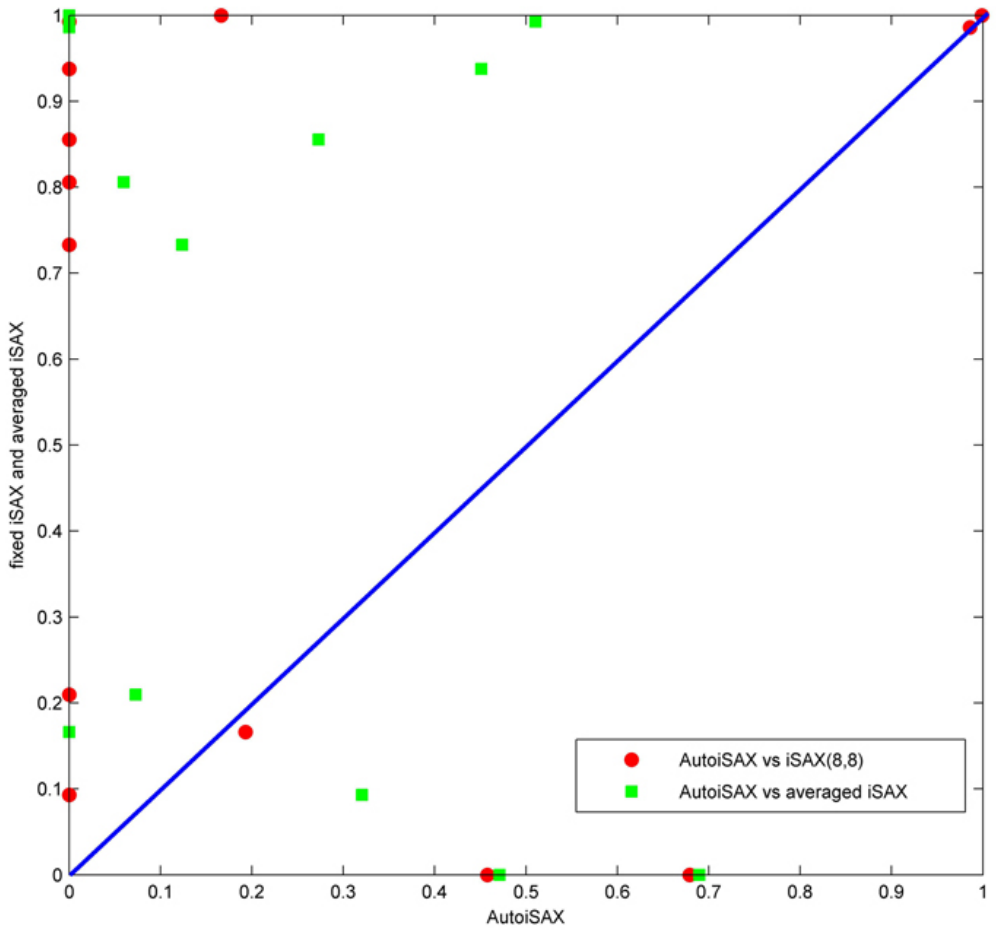


Figure 6.8: Inter-motif normalized distances. Circles: AutoiSAX vs iSAX(8,8). Squares: AutoiSAX vs average iSAX. Above the line the distance is smaller (better).

dataset benchmark (aggregated in (Castro and Azevedo, 2011)) is selected for the experiment. The question we would like to be able to answer is whether the derived motifs are more meaningful, as we are using two further implicit dimensions to filter motifs: complexity and standard deviation. To answer the question the average Euclidean Distance between each match in the dataset is measured. Matching time series should present a small ED for the motif to be interesting. In Figure 6.8 we show the scatter-plot of the pairs of average intra-motif distance for the approaches. Results are normalized in the $[0, 1]$ interval. Distances are only shown for the datasets motifs were found.

It can be observed that the intra-motif distance of the AutoiSAX approach is smaller than the other two approaches', for most datasets. This is expected, as

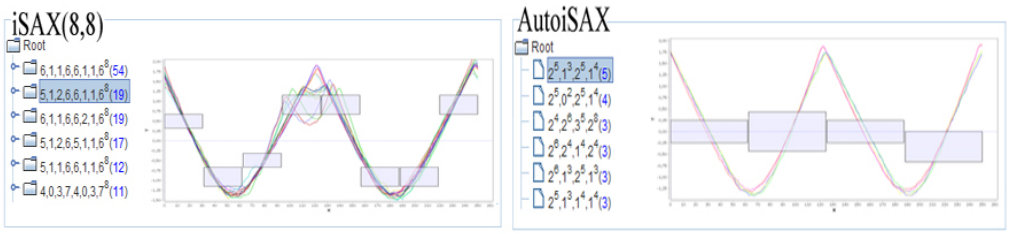


Figure 6.9: Motifs extracted from the projectile shapes dataset using the iSAX(8, 8) (top) and AutoiSAX (bottom) approaches, using the iMotifs Visualization Tool.

AutoiSAX adds two other dimensions (besides the average) to further restrict the motifs: complexity and standard deviation. If the motifs size and alphabet size are not previously fixed, their natural length and variability will cause matches to be more accurate. For example, let us consider two iSAX(8,8) words with standard deviations of 0.1 and 1.2. If by chance their mean is the same, they would match with fixed iSAX, but would generate two very different words using AutoiSAX. Using our approach, only very similar time series can match. To investigate whether this also affects the number of discovered motifs, we count the number of motifs returned for each approach and dataset. For the majority of the datasets the number of returned motifs is smaller using AutoiSAX approach. This suggests that using AutoiSAX to extract symbolic motifs can act as a filter to prune spurious motifs. The effect of using our approach in motif mining is shown with a representative example in Figure 6.9. The iMotifs Visualization Tool (Castro, 2011a) is used to visualize the extracted motifs from the projectiles shape dataset from Yankov et al. (2007a) using the fixed iSAX and AutoiSAX approaches.

The first thing to notice is that the number of extracted motifs (enclosed in parentheses) is larger in the fixed iSAX approach. Also, some incorrect motifs are found with this approach. For instance, the motif in Figure 6.9 at the fourth and fifth symbols is a false discovery. As only the mean is used to match time series, the approach is unable to prune out motifs with a different standard deviation. The AutoiSAX approach exploits more information on the series to obtain fewer and more similar matches. This suggests AutoiSAX can be used as a parameter-free alternative to mine symbolic motifs.

6.5.5 Discussion

The major advantage of AutoiSAX is estimating the symbolic length and alphabet size parameters for the iSAX representation. As a consequence of the techniques used to obtain these parameters, two more dimensions become available (for free) for analyzing the represented time series. First, complexity estimation allows to assign more symbols to complex time series. Second, standard deviation computation permits to adjust the representation's resolution. Furthermore, it enables different resolutions within the same symbolic word. This is a major feature, as time series are typically long and present large variations in behavior along their length. Even small time series can have wildly changing behavior, which a single resolution per-series simply cannot capture. This fully exploits iSAX's multiresolution to applications beyond indexing. In this task, interleaving between different resolutions is performed by the indexing algorithm. The resolutions are doubled whenever the count of a time series container surpasses a given threshold (Shieh and Keogh, 2008). In other mining tasks, such as visualization, classification and motif discovery, this was not previously possible as the only way to use intra-word multiresolution was to manually set the iSAX word's parameters. One potential feature of our approach is that it seems to be more suitable to deal with extreme values than classical iSAX. The standard deviation calculation appears to compensate for the information loss caused by average smoothing. These two dimensions provide further information for visualization, classification and motif discovery algorithms, which are leveraged to obtain more accurate results, as we have seen in section 5. In visualization, one can get a more intuitive grasp at the original series shape just by looking at the representation. This is relevant as domain experts can more effectively visually detect interesting patterns in their data using tools such as the aforementioned iMotifs (Castro, 2011a). One can also obtain better classification accuracy. It seems that this measure is highly correlated with the resolution used. However, too high resolutions can be harmful as we have shown. AutoiSAX appears to provide a nice trade-off between the minimum resolution required to discriminate between the classes and a not too high resolution that can overfit the data. Finally,

more interesting motifs can be found as closer matches are retrieved and spurious motifs discarded. The addition of complexity and deviation information in the motif computation seems to impose an important filter in the process. It remains to be investigated whether these motifs are also statistically more significant by using our motifs statistical significance approach, tailored for fixed iSAX (Castro and Azevedo, 2011). It also remains as future work to research whether a more efficient indexing algorithm can be obtained by mixing AutoiSAX in the original iSAX indexing framework. Preliminary results with the benchmark datasets (Castro and Azevedo, 2011) show that the Tightness of Lower Bound measure depends highly on the resolution at hand, as expected. Nevertheless, further experimentation is required to reach a firm conclusion whether the measure is appropriate in the current settings. We are also currently investigating whether the application of AutoiSAX to streaming data is trivial or requires more elaboration. Finally, we stress out that almost all these mining techniques can effectively become parameter-free as a consequence of using our approach.

6.5.6 Conclusion

We have proposed an approach to automatically estimate the parameters of the iSAX time series representation. The novelty of our work is that it enables efficient computation of the symbolic length and alphabet size parameters. Furthermore, it permits to calculate the alphabet size for each symbol within the same time series. The approach, referred as AutoiSAX, is based on two very simple ideas. First, complex time series require larger symbolic lengths to better capture their behavior. Second, as time series segments present different deviations from the mean, they should be represented using different alphabet sizes. Our technique allows to leverage iSAX's intra-word multiresolution, which effectively brings iSAX to other tasks besides time series indexing. We have shown that AutoiSAX improves visualization interpretability, classification accuracy and motif mining results. We also aim to highlight the importance of making iSAX a parameter-free approach, as the large number of researchers and practitioners using it can benefit from it.

Reproducibility Note

All experiments, source code and datasets are available online at Castro (2012).

- 7 -

Conclusion

In the beginning of this thesis we define our research statement:

We believe that the state of the art in motif discovery can be improved, by developing scalable algorithms with fewer parameters. To further increase the understanding in time series mining, we aim to develop a methodology to automatically evaluate the discovered motifs. We hope that this approach has significant impact in the time series data mining community.

To thoroughly conclude this thesis we will evaluate the achievements in relation to the above statement.

The lack of a scalable algorithm to mine frequent motifs in times series databases is an importante research opportunity. Since the amount of present databases is typically large, quadratic algorithms are not sufficiently efficient. For example, one quadratic exact motif discovery algorithm can take days to complete execution, for a relatively moderate dataset. This motivation has led us to create the MrMotif algorithm (section 4). It is an approximate algorithm, as it is our strong conviction that the trade-off between accuracy and execution time favors the latter. Its time-efficiency comes from using an approximate time series representation, constant access time data structures (hash tables) and leveraging iSAX's multiresolution property to skip expensive distance calculations in the raw data. It is disk-efficient since it performs one single disk scan. It is also a strong candidate for mining streaming data. The combination with the existing Space-Saving algorithm yields a space-efficient procedure. It can also be applied to multivariate datasets, requires the setting of one single parameter and is robust to noise. MrMotif has been receiving

an interesting support by the time series data mining community. We have received about 90 requests for the algorithm's source code. It has been used and studied in several academic, industrial and government institutions. Currently it has received about ten citations in just two years. Its simplicity and scalability have been pointed as its best features. The ability to visualize the extracted motifs using the iMotifs Tool has been described as a great advantage of the algorithm.

To the best of our knowledge, there is no approach in the literature to effectively evaluate the motifs extracted by any motif discovery algorithm. This is important because motifs are evaluated in a subjective way by humans. The prohitively large number of motifs typically returned by motif discovery algorithms lead any effort to failure, as many of these motifs are caused by chance and are therefore meaningless. In this thesis, the motif evaluation problem is framed as the computation of a motif's statistical significance (chapter 5). In practice, we compute each motif's p-value. Our innovation builds up on existing work in time series representations, motif discovery algorithms and proposals from other areas such as bioinformatics. It is essentially parameter-free as the significance level is automatically derived. Also, it borrows this property to motif discovery algorithms, as no Top-K or support parameter are required. It can act as a filter which significantly reduces the number of returned motifs. As a consequence of our approach, a ranking of motifs is obtained by considering their statistical significance. We believe that our approach provides researchers and practitioners with an important technique to evaluate the degree of relevance of each extracted motif. We also aim to highlight the importance of evaluating motifs since it is crucial to make motif mining an useful task in practice. This proposal received a Best Student Paper Award at the SIAM SDM'2011.

Finally, we attempt to provide fundamental research on a crucial area in time series data mining which is extensively used throughout this thesis – time series representations. We aim to improve the state of the art iSAX time series representation by proposing an approach to automatically estimate its parameters (chapter 6). The need to set two parameters limits iSAX's applicability, since they are highly dependent on the domain of application. To set them to a fixed value can

provide non-optimal results. To experimentally optimize them can be unfeasible, as datasets are typically large. We believe that providing iSAX as a parameter-free approach can be important for the large number of iSAX users described in the literature. Our approach is fast and intuitive. It is based on the simple ideas of time series complexity and deviation. We have shown that using AutoiSAX can improve visualization interpretability, classification accuracy and motif mining results. These achievements can be explained by the incorporation of two extra dimensions: complexity and deviation. These dimensions are available without requiring any extra space in the mining process.

Our proposals are biased towards approximate approaches, which is a direct consequence of our belief that quadratic approaches make motif mining hard to use and adopt in practice. In a real scenario, for instance where a medical doctor needs to be convinced that a mining approach can help him reduce the ECG analysis time, scalable approaches are a requirement.

The extensive use of the iSAX representation throughout this thesis is supported by its generality, interpretability, and scalability. One of our main philosophical views is that data mining should be strongly connected with the user. Thus, intuitive and interpretable approaches are clearly the way to go. In our opinion, time series representations and in particular iSAX can have a crucial role in this endeavor. To be able to provide an intuition to the user about the shape of a time series of length 1 million with just one word of 8 symbols is a great advantage.

All in all, we believe we have contributed to the state of the art in time series motif mining: scalable and parameter-free approaches are the directions to follow to make motif mining meaningful in practice.

References

Agrawal, R., Psaila, G., Wimmers, E., and Zaït, M. (1995). Querying shapes of histories. In *Proceedings of the 21th International Conference on Very Large Data Bases*, pages 502–514. Morgan Kaufmann Publishers Inc.

Ahmadi, S., Padoy, N., Rybachuk, K., Feussner, H., Heinin, S., and Navab, N. (2009). Motif discovery in or sensor data with application to surgical workflow analysis and activity detection. In *M2CAI workshop, MICCAI, London*. Citeseer.

Alcock, C., Allsman, R., Alves, D., Axelrod, T., Becker, A., Bennett, D., Cook, K., Freeman, K., Griest, K., Lehner, M., et al. (1997). Macho project publications. *ApJ*, 482:89.

Androulakis, I. (2005). New approaches for representing, analyzing and visualizing complex kinetic mechanisms. *Computer Aided Chemical Engineering*, 20:235–240.

Androulakis, I., Wu, J., Vitolo, J., and Roth, C. (2005). Selecting maximally informative genes to enable temporal expression profiling analysis. *Proc. of Foundations of Systems Biology in Engineering*.

Argyros, T. and Ermopoulos, C. (2003). Efficient subsequence matching in time series databases under time and amplitude transformations. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 481, Washington, DC, USA. IEEE Computer Society.

Azevedo, P., Silva, C., Rodrigues, J., Loureiro-Ferreira, N., and Brito, R. (2005). Detection of hydrophobic clusters in molecular dynamics protein unfolding simulations using association rules. *Biological and Medical Data Analysis*, pages 329–337.

Batista, G., Wang, X., and Keogh, E. (2011). A complexity-invariant distance

measure for time series. In *SDM-2011: Proceedings of SIAM International Conference on Data Mining, Philadelphia, PA, USA*.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300.

Benjamini, Y. and Leshno, M. (2005). Statistical methods for data mining. *Data Mining and Knowledge Discovery Handbook*, pages 565–587.

Berndt, D. J. and Clifford, J. (1996). Finding patterns in time series: a dynamic programming approach. *Advances in knowledge discovery and data mining*, pages 229–248.

Boeva, V., Clément, J., Régnier, M., Roytberg, M., and Makeev, V. (2007). Exact p-value calculation for heterotypic clusters of regulatory motifs and its application in computational annotation of cis-regulatory modules. *Algorithms for molecular biology*, 2(1):13.

Buhler, J. and Tompa, M. (2002). Finding Motifs Using Random Projections. *Journal of Computational Biology*, 9(2):225–242.

Buza, K. and Schmidt-Thieme, L. (2010). Motif-based classification of time series with bayesian networks and svms. *Advances in Data Analysis, Data Handling and Business Intelligence*, pages 105–114.

Camerra, A., Palpanas, T., Shieh, J., and Keogh, E. (2010). isax 2.0: Indexing and mining one billion time series. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 58–67. IEEE.

Castro, N. (2010). Multiresolution motif discovery in time series website. `http://www.di.uminho.pt/~castro/mrmotif`.

Castro, N. (2011a). iMotifs: Interactive Time Series Motif Discovery and Visualization Tool. `http://www.di.uminho.pt/~castro/imotifs`.

Castro, N. (2011b). Time series motifs statistical significance website. `http://www.di.uminho.pt/~castro/stat`.

Castro, N. (2012). Autoisax website. `http://www.di.uminho.pt/~castro/autoisax`.

Castro, N. and Azevedo, P. (2010). Multiresolution Motif Discovery in Time Series. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, 2010, Columbus, Ohio, USA*, pages 665–676. SIAM.

Castro, N. and Azevedo, P. (2011). Time Series Motifs Statistical Significance. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2011, 2011, Mesa, Arizona, USA*, pages 687–698. SIAM.

Castro, N. C. and Azevedo, P. J. (2012a). Parameter-free iSAX. (under peer review).

Castro, N. C. and Azevedo, P. J. (2012b). Significant motifs in time series. *Statistical Analysis and Data Mining*, 5(1):35–53.

Catalano, J., Armstrong, T., and Oates, T. (2006). Discovering patterns in real-valued time series. *Proc. of the Tenth European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD), Septembar*.

Celly, B. and Zordan, V. (2004). Animated people textures. In *Proc. of 17th International Conference on Computer Animation and Social Agents (CASA)*. Citeseer.

Chakrabarti, K., Keogh, E., Mehrotra, S., and Pazzani, M. (2002). Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228.

Chan, K.-P. and Fu, A. W.-C. (1999). Efficient time series matching by wavelets. In *ICDE*, pages 126–133.

Chiu, B., Keogh, E., and Lonardi, S. (2003). Probabilistic discovery of time series motifs. In *KDD '03: Proceedings of the 9th ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, pages 493–498, New York, NY, USA. ACM.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning (Historical Archive)*, 20(3):273–297.

Das, G., Gunopulos, D., and Mannila, H. (1997). Finding similar time series. In *PKDD '97: Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 88–100, London, UK. Springer-Verlag.

Dasgupta, D. and Forrest, S. (1996). Novelty detection in time series data using ideas from immunology. In *Proceedings of the International Conference on Intelligent Systems*, pages 82–87.

Denton, A., Besemann, C., and Dorr, D. (2009). Pattern-based time-series subsequence clustering using radial distribution functions. *Knowledge and Information Systems*, 18(1):1–27.

Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., and Keogh, E. (2008). Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552.

Emonet, R., Varadarajan, J., and Odobez, J. (2011). Extracting and locating temporal motifs in video scenes using a hierarchical non parametric bayesian model. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, June 2011.*

Esling, P. and Agon, C. (2011). Time series data mining and analysis. *ACM Computing Surveys (to appear).*

Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. In *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, pages 419–429, New York, NY, USA. ACM.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., et al. (1996). Knowledge Discovery and Data Mining: Towards a Unifying Framework. *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR*, pages 82–88.

Ferreira, P. G. and Azevedo, P. J. (2007). Evaluating protein motif significance measures: A case study on prosite patterns. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)*, pages 171–178.

Ferreira, P. G., Azevedo, P. J., Silva, C. G., and Brito, R. M. M. (2006). Mining approximate motifs in time series. In *Discovery Science*, pages 89–101.

Fuchs, E., Gruber, T., Nitschke, J., and Sick, B. (2009). On-line motif detection in time series with swiftmotif. *Pattern Recognition*, 42(11):3015–3031.

Futschik, M. and Carlisle, B. (2005). Noise-robust soft clustering of gene expression time-course data. *Journal of bioinformatics and computational biology*, 3(4):965–988.

Gama, O., Carvalho, P., Afonso, J., and Mendes, P. (2009). An improved mac protocol with a reconfiguration scheme for wireless e-health systems requiring quality of service. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 582–586. IEEE.

Gao, Y., He, J., Zou, J., Zeng, R., and Liang, X. (2004). Fractal simulation of soil breakdown under lightning current. *Journal of electrostatics*, 61(3-4):197–207.

Geurts, P. (2001). Pattern extraction for time series classification. *Principles of Data Mining and Knowledge Discovery*, pages 115–127.

Grass, J. and Zilberstein, S. (1996). Anytime algorithm development tools. *ACM SIGART Bulletin*, 7(2):20–27.

Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I., and Isbell, C. (2005). Unsupervised activity discovery and characterization from event-streams. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI05)*. Citeseer.

Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

Hanhijärvi, S., Puolamäki, K., and Garriga, G. (2009). Multiple Hypothesis Testing in Pattern Discovery. *stat*, 1050:29.

He, H. and Singh, A. (2006). Graphrank: Statistical modeling and mining of significant subgraphs in the feature space. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 885–890.

Hollunder, J., Friedel, M., Beyer, A., Workman, C., and Wilhelm, T. (2007). DASS: efficient discovery and p-value calculation of substructures in unordered data. *Bioinformatics*, 23(1):77.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Hu, B., Rakthanmanon, T., Hao, Y., Evans, S., Lonardi, S., and Keogh, E. (2011). Discovering the intrinsic cardinality and dimensionality of time series using mdl. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 1086–1091. IEEE.

Jacquemont, S., Jacquenet, F., and Sebban, M. (2009). Mining probabilistic automata: a statistical view of sequential pattern mining. *Machine Learning*, 75(1):91–127.

Jensen, K., Styczynski, M., Rigoutsos, I., and Stephanopoulos, G. (2005). A generic motif discovery algorithm for sequential data. *Bioinformatics*, 22(1):21.

Kalpakis, K., Gada, D., and Puttagunta, V. (2001). Distance measures for effective clustering of arima time-series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 273–280. IEEE.

Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286.

Keogh, E. and Folias, T. (2002). The UCR Time Series Data Mining Archive. Riverside CA. *University of California-Computer Science & Engineering Department.*

Keogh, E. and Kasetty, S. (2003). On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371.

Keogh, E. and Lin, J. (2005). Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowl. Inf. Syst.*, 8(2):154–177.

Keogh, E., Lin, J., and Fu, A. (2005). HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, page 233. IEEE Computer Society.

Keogh, E., Lonardi, S., and Chiu, B. (2002). Finding surprising patterns in a time series database in linear time and space. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 550–556.

Keogh, E., Palpanas, T., Zordan, V. B., Gunopulos, D., and Cardle, M. (2004). Indexing large human-motion databases. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 780–791. VLDB Endowment.

Keogh, E., Xi, X., Wei, L., and Ratanamahatana, C. (2009). Ucr time series classification/clustering page. *Training and testing data sets: Available online: http://www. cs. ucr. edu/eamonn/time series data/(accessed on 18 July 2011).*

Keogh, E. J. (2006). A decade of progress in indexing and mining large time series databases. In *VLDB*, page 1268.

Keogh, E. J., Lonardi, S., Ratanamahatana, C. A., Wei, L., Lee, S.-H., and Handley, J. (2007). Compression-based data mining of sequential data. *Data Min. Knowl. Discov.*, 14(1):99–129.

Keogh, E. J. and Pazzani, M. J. (1999). Relevance feedback retrieval of time series data. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 183–190, New York, NY, USA. ACM.

Kim, S., Park, S., and Chu, W. (2001). An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases. *Proceedings of the 17th International Conference on Data Engineering table of contents*, pages 607–614.

Knorr, T., Schmidt-Thieme, L., and Johner, C. (2009). Identifying patients at risk. *Cooperation in Classification and Data Analysis*, pages 131–140.

Korn, F., Jagadish, H. V., and Faloutsos, C. (1997). Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 289–300, New York, NY, USA. ACM.

Lam, H., Pham, N., and Calders, T. (2011). Online Discovery of Top-k Similar Motifs in Time Series Data. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2011, 2011, Mesa, Arizona, USA*. SIAM.

Li, Y. and Lin, J. (2010). Approximate variable-length time series motif discovery using grammar inference. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, pages 1–9. ACM.

Lin, J., Keogh, E., Lonardi, S., and Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, page 11. ACM.

Lin, J., Keogh, E., Lonardi, S., Lankford, J., and Nystrom, D. (2004). Visually mining and monitoring massive time series. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 460–469. ACM.

Lin, J., Keogh, E., Lonardi, S., and Patel, P. (2002). Finding motifs in time series. In *Proceedings of the Second Workshop on Temporal Data Mining*, Edmonton, Alberta, Canada.

Lin, J., Keogh, E., Wei, L., and Lonardi, S. (2007). Experiencing SAX: a novel symbolic representation of time series. *Data mining and knowledge discovery*, 15(2):107–144.

Lkhagva, B., Suzuki, Y., and Kawagoe, K. (2006). New Time Series Data Representation ESAX for Financial Applications. *Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06)-Volume 00*.

Low Kam, C., Mas, A., and Teisseire, M. (2000). Mining for unexpected sequential patterns given a Markov model. Unpublished.

Marschall, T. and Rahmann, S. (2009). Efficient exact motif discovery. *Bioinformatics*, 25(12):i356.

Matias, C., Schbath, S., Birmelé, E., Daudin, J., and Robin, S. (2006). Network motifs: mean and variance for the count. *REVSTAT–Statistical Journal*, 4(1):31–51.

McGovern, A., Rosendahl, H., Kruger, A., Beaton, M., Brown, R., and Droegemeier, K. (2007). Anticipating the formation of tornadoes through data mining. In *Fifth Conference on Artificial Intelligence Applications to Environmental Science*.

Metwally, A., Agrawal, D., and Abbadi, A. (2005). Efficient computation of frequent and top-k elements in data streams. *Database Theory-ICDT 2005*, pages 398–412.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. (2002). Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824.

Minnen, D. (2008). *Unsupervised discovery of activity primitives from multivariate sensor data*. PhD thesis, Georgia Institute of Technology.

Minnen, D., Isbell, C., Essa, I., and Starner, T. (2007a). Detecting Subdimensional Motifs: An Efficient Algorithm for Generalized Multivariate Pattern Discovery. *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 601–606.

Minnen, D., Isbell, C., Essa, I., and Starner, T. (2007b). Discovering multivariate motifs using subsequence density estimation and greedy mixture learning. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 615. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Minnen, D., Starner, T., Essa, I., and Isbell, C. (2007c). Improving Activity Discovery with Automatic Neighborhood Estimation. *Int. Joint Conf. on Artificial Intelligence*.

Minnen, D., Starner, T., Essa, M., and Isbell, C. (2007d). Discovering characteristic actions from on-body sensor data. In *Wearable Computers, 2006 10th IEEE International Symposium on*, pages 11–18. IEEE.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

Mohammad, Y. (2009). Constrained Motif Discovery in Time Series. *New Generation Computing*, 24(1):319–346.

Mörchen, F. and Ultsch, A. (2007). Efficient mining of understandable patterns from multivariate interval time series. *Data Mining and Knowledge Discovery*, 15(2):181–215.

Mueen, A. and Keogh, E. (2010). Online discovery and maintenance of time series motifs. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1089–1098. ACM.

Mueen, A., Keogh, E., and Bigdely-Shamlo, N. (2009a). Finding time series motifs in disk-resident data. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 367–376. IEEE.

Mueen, A., Keogh, E., Zhu, Q., Cash, S., and Westover, B. (2009b). Exact discovery of time series motifs. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2009)*, pages 473–484. Citeseer.

Muscariello, A., Gravier, G., and Bimbot, F. (2011). An efficient method for the unsupervised discovery of signalling motifs in large audio streams. In *International Workshop on Content-Based Multimedia Indexing*.

Nuel, G. (2006). Effective p-value computations using Finite Markov Chain Imbedding(FMCI): application to local score and to pattern statistics. *Algorithms for Molecular Biology*, 1(1):5.

Oates, T. (2002). PERUSE: An unsupervised algorithm for finding recurring patterns in time series. *Proceedings of the IEEE International Conference on Data Mining*, pages 330–337.

Ouyang, R., Ren, L., Cheng, W., and Zhou, C. (2010). Similarity search and pattern discovery in hydrological time series data mining. *Hydrological Processes*, 24(9):1198–1210.

Penedones, H. (2005). *Data Mining in Telecommunication Databases – Anomaly detection*. Siemens Portugal and University of Porto.

Pham, N., Le, Q., and Dang, T. (2010). Two novel adaptive symbolic representations for similarity search in time series databases. In *Web Conference (APWEB), 2010 12th International Asia-Pacific*, pages 181–187. IEEE.

Piatetsky-Shapiro, G. (2006). Data mining course. Slides available at the URL *http : //www.kdnuggets.com/data_mining_course/index.html*.

Piatetsky-Shapiro, G. and Frawley, W. J., editors (1991). *Knowledge Discovery in Databases*. AAAI/MIT Press.

Picard, F., Daudin, J., Koskas, M., Schbath, S., and Robin, S. (2008). Assessing the exceptionality of network motifs. *Journal of Computational Biology*, 15(1):1–20.

Ratanamahatana, C. and Keogh, E. (2005). Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining (SDM'05)*, pages 506–510. Citeseer.

Rebbapragada, U. (2007). Introduction to machine learning: Research on time series.

Régnier, M. and Vandenbogaert, M. (2006). Comparison of statistical significance criteria. *Journal of Bioinformatics and Computational Biology*, 4(2):537–552.

Ribeca, P. and Raineri, E. (2008). Faster exact Markovian probability functions for motif occurrences: a DFA-only approach. *Bioinformatics*, 24(24):2839.

Robin, S., Rodolphe, F., and Schbath, S. (2005). *DNA, words and models*. Cambridge Univ Pr.

Robin, S. and Schbath, S. (2001). Numerical comparison of several approximations of the word count distribution in random sequences. *Journal of Computational Biology*, 8(4):349–359.

Robin, S., Schbath, S., and Vandewalle, V. (2007). Statistical tests to compare motif count exceptionalities. *BMC bioinformatics*, 8(1):84.

Rombo, S. and Terracina, G. (2004). Discovering representative models in large time series databases. *Flexible Query Answering Systems*, pages 84–97.

Santosh Bangalore, S., Wang, J., and Allison, D. (2009). How accurate are the extremely small p-values used in genomic research: An evaluation of numerical libraries. *Computational statistics & data analysis*, 53(7):2446–2452.

Schbath, S. (2000). An overview on the distribution of word counts in Markov chains. *Journal of Computational Biology*, 7(1-2):193–201.

Schbath, S. (2006). Statistics of motifs. *Atelier de formation*, 1502.

Shieh, J. and Keogh, E. (2008). iSAX: indexing and mining terabyte sized time series. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631.

Storey, J. and Tibshirani, R. (2003). Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences of the United States of America*, 100(16):9440.

Tanaka, Y., Iwamoto, K., and Uehara, K. (2005). Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle. *Machine Learning*, 58(2):269–300.

Tang, H. and Liao, S. (2008). Discovering original motifs with different lengths from time series. *Knowledge-Based Systems*, 21(7):666–671.

Tufte, E. and Howard, G. (1983). *The visual display of quantitative information*, volume 16. Graphics Press Cheshire, CT.

Udechukwu, A., Barker, K., and Alhajj, R. (2004). Discovering all frequent trends in time series. In *Proceedings of the winter international synposium on Information and communication technologies*, pages 1–6. Trinity College Dublin.

Vahdatpour, A., Amini, N., and Sarrafzadeh, M. (2009). Toward unsupervised activity discovery using multi-dimensional motif detection in time series. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1261–1266. Morgan Kaufmann Publishers Inc.

Vlachos, M., Lin, J., Keogh, E., and Gunopulos, D. (2003). A wavelet-based anytime algorithm for k-means clustering of time series. In *In Proc. Workshop on Clustering High Dimensionality Data and Its Applications*. Citeseer.

Wang, L., Chng, E., and Li, H. (2010). A tree-construction search approach for multivariate time series motifs discovery. *Pattern Recognition Letters*, 31(9):869–875.

Wang, P., Wang, H., and Wang, W. (2011). Finding semantics in time series. In *Proceedings of the 2011 international conference on Management of data*, pages 385–396. ACM.

Webb, G. (2007). Discovering significant patterns. *Machine Learning*, 68(1):1–33.

Whitehorn, M. (2010). The parable of the beer and diapers. `http://www.theregister.co.uk/2006/08/15/beer_diapers/`.

Wilson, W., Birkin, P., and Aickelin, U. (2008). The motif tracking algorithm. *International Journal of Automation and Computing*, 5(1):32–44.

Yankov, D., Keogh, E., Medina, J., Chiu, B., and Zordan, V. (2007a). Detecting time series motifs under uniform scaling. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 844–853. ACM.

Yankov, D., Keogh, E., and Rebbapragada, U. (2007b). Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. In *7th IEEE International Conference on Data Mining*, pages 381–390.

Ye, L., Wang, X., Yankov, D., and Keogh, E. J. (2008). The asymmetric approximate anytime join: A new primitive with applications to data mining. In *SDM*, pages 363–374.

Zhang, H., Padmanabhan, B., and Tuzhilin, A. (2004). On the discovery of significant statistical quantitative rules. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383. ACM.

Zhang, J., Jiang, B., Li, M., Tromp, J., Zhang, X., and Zhang, M. (2007). Computing exact P-values for DNA motifs. *Bioinformatics*, 23(5):531.